

INSIDE: ERIC PINNELL REVIEWS COMDEX

OS/2 Monthly

The Independent Guide to OS/2 Computing

Issue Seven / \$4.50

REXX

A Modern Day King

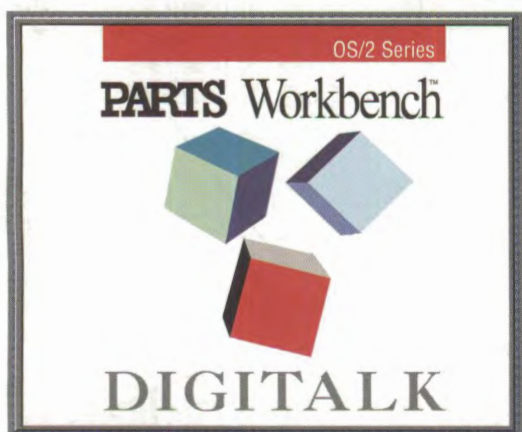
PLUS

REXX And The GUI

Steve Price on REXX

Les Bell Interviews Mike Kogan

Parts Review





The Only Tool ... to create, modify, You'll Ever Need manage, browse and

translate resources such as dialog boxes, bitmaps, icons, cursors and menus.

Increase the productivity of your development work by organizing resources into projects and seamlessly integrate all the resource editors.

Reduce the amount of time required to design the user interface of your application—use the state of the art visual editors to interactively design and modify dialogs, bitmaps, icons, cursors, menus, accelerators, string tables and all other resources—or just use any of the hundreds of professionally designed resources included.

Capture images from anywhere on the screen directly into the image editor. Create bitmaps, icons and cursors from the captured image.

Customize the look and feel of your applications—directly extract, modify or replace any resource in any program file, EXE, DLL—no source code is needed.



Translate applications into
foreign languages without

having access to the applications source code—

directly modify all messages, menus and dialogs into any national language.

Edit, copy and paste resources directly into and from any EXE, RES, DLL, RC program file. Eliminate the time consuming edit-compile-link cycle—increasing your productivity dramatically.

Migrate your applications to new 32-bit OS/2 2.0 and NT applications utilizing the transparent and automatic resource translation capabilities. Create your resources once on any platform and instantly use them on any other platform.

Project views and preview
browsers allow you to efficiently
manage your resources.



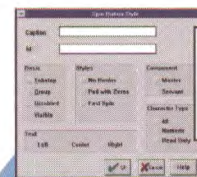
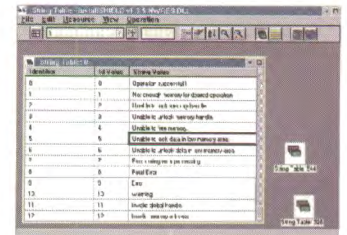
Quickly design complex
dialogs using powerful
design tools in the
Dialog Editor.

The Menu Editor.

Create bitmaps, icons,
cursors using the
advanced image editors.



Find, replace and translate
text messages in string tables.



MICROSOFT
WINDOWS.

**Translate
all your
resources**



MICROSOFT
WINDOWS NT.



The Stirling Group
Making it easy to make™

172 OLD MILL DRIVE • SCHAMBURG, IL 60193 • USA **1-800-3 SHIELD**
1-800-374-4353

CALL 708-307-9197 • FAX 708-307-9340
CompuServe: GO STIRLING • MCI Mail: STIRLING

Introducing PM/FOCUS



The Power of Graphical Development In an Exciting New Environment

Over three years of development and testing have transformed a vision into reality. An ideal development environment created by the perfect harmony of proven database technology within an exciting new visual landscape. Information Builders introduces PM/FOCUS, a graphical development and decision support system combining unparalleled data access and reporting with the versatility, stability and power of OS/2 2.0.

PM/FOCUS offers a rich graphical tool set complete with File Painter, Forms Painter, and Query Painter to build even the most complex applications without writing code. Application components such as databases, forms, and procedures, become simple graphic objects that can be manipulated and activated with a point, click, and drag of your mouse. The comprehensive array of list boxes, check boxes, multiple fonts, radio buttons and other graphical

images, offer an intuitive interface that's remarkably attractive and easy to use. PM/FOCUS comes with its own proven database engine. Optional interfaces let you create sensational GUI applications for most other popular SQL databases. And PM/FOCUS is EDA/SQL enabled. This means you can have access to over 50 relational and non-relational file types on more than 35 different operating platforms.

PM/FOCUS. It could well be the reason why OS/2 was invented. Call your local Information Builders branch office or Micro Products Telesales at **1-800-969-INFO.** FAX (212) 695-3247.

IBI PM/FOCUS
Information Builders, Inc.
1250 Broadway, New York, NY 10001

FaxWorks™

The



*fax
solution*

The Right Tool For The Job

Before FaxWorks OS/2, you had to try to use a DOS or Windows based fax program to fax from OS/2. Not anymore! FaxWorks is a true OS/2 program. It uses the advanced features of OS/2 to turn your computer into a sophisticated send/receive fax machine. By taking advantage of OS/2's true multitasking capability, FaxWorks allows you to send, receive and print faxes in the background. So your computer can be doing more important things than just sending a fax. FaxWorks works the way you want.

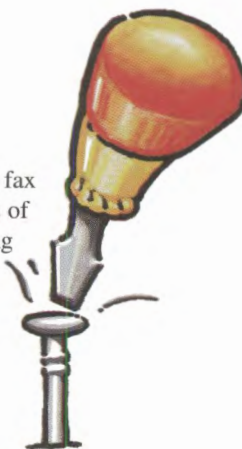
Fax Software for OS/2

FaxWorks, the OS/2 fax solution, makes it easy to send and receive faxes on your PC. With FaxWorks and a fax modem, faxing from OS/2 is as easy as printing. Since FaxWorks includes an OS/2 printer driver, you can fax from your Windows and DOS applications running under OS/2. FaxWorks also supports Adobe Type Manager, so the faxes you send will have that "just-printed" look. FaxWorks, it meets your needs.

All SofNet products are
backed by a 60-day
money-back
guarantee.



FaxWorks is a trademark of SofNet, Inc. All other referenced products are trademarks or registered trademarks of their respective manufacturers.



Support For Your Fax Modem

FaxWorks works with the fax modems you want to use, like Intel, Class 1 and Class 2. This gives you the flexibility you want in selecting a fax modem. Support for more fax modems, just another reason why you need FaxWorks.

SOFTNET

The Standard in Fax Software

380 Interstate North Pkwy. • Suite 150 Atlanta, Georgia 30339
(404) 984-8088 Fax 984-9956

1-800-4FAXWORKS

CONTENTS

FEATURES

REXX: A MODERN DAY KING

Eloy O. Cruz-Bizet

16

CALLING REXX

Steve Price

19

LES BELL INTERVIEWS MIKE KOGAN

Les Bell

21

AN OS/2-PM EDITOR USING MLE

Brian R. Anderson

29

DEPARTMENTS

PUBLISHER'S NOTES 2

Joel Siragher

EDITOR'S CORNER 3

Brett Kotch

BEGINNER'S SLOPE 4

Bill Zinsmeyer

DISCOVERING THE WORKPLACE SHELL 8

Brett Kotch

FOR IMMEDIATE RELEASE 9

INDEX OF ADVERTISERS 15

THE ULTIMATE OS/2 GAME 43

Paint By Numbers . . . Timur Tabi

OBJECT OBJECTIVE 46

Sources of Class Libraries . . . David Moskowitz

IN THE TRENCHES 48

Eric Pinnel

REVIEW 11

HyperAccess/5 • Parts Workbench for OS/2 • COMDEX

Ron Beauchemin and Eric Pinnel

PUBLISHER'S NOTES

Publisher

Joel Siragher

Editor

Brett Kotch

Contributing Editors

Les Bell, David Moskowitz, Eric Pinnel, Guy Scharf,
Timur Tabi, Ron Beauchemin

Technical Review Board

Noel Bergman, Mel Hallerman

Advertising Director

Rachel Siragher

Graphical Director

Deborah S. Weightman

Volume 1, Issue 7, April 1993

OS/2 Monthly is published monthly by JDS Publishing, P.O. Box 4351, Highland Park, NJ 08904 908-937-6542.

OS/2 Monthly is published independently of IBM Corp., which is not responsible in any way for editorial policy or other content of this publication.

Postmaster: There is a second class subscriber mailing permit. The original entry point is Kilmer Mailing facility in New Brunswick, NJ, with an additional entry pending in Hartford, CT. Address corrections requested.

Subscription rate for the U.S. is \$39 for 12 issues. Canadian and Mexican are \$45. All other foreign subscriptions are \$87. Visa/Mastercard accepted.

OS/2 Monthly welcomes unsolicited submission of written material and artwork; however, no responsibility for such submissions can be assumed by the publisher in any event.

OS/2 Monthly is a trademark of JDS Publishing. All rights to contents of this publication are reserved, and nothing may be reproduced from it in whole or part without first obtaining permission in writing from the publisher.

Articles and items appearing in OS/2 Monthly do not necessarily reflect the views and opinion of JDS Publishing, its staff, or organization.

All trademarks used herein are property of their respective holders.

OS/2 is a registered trademark of IBM Corporation and is used by JDS Publishing under license.

It was a dark and stormy night. It was the kind of night you would read about in one of those old detective stories, and never forget. I decided to go down to the local watering hole for a drink, maybe a little action.

There it was. "Abies Place". It had the characteristic flickering neon sign that should have died years ago. My old pal Abie the snail, was tending. He said 'the usual, Mr Siragher?'

"Yeah, Abie, the usual" Already, I was suspicious. He looked kinda worried. His eyes never left me as he reached under the counter. I grabbed his hand before he could do something he'd regret.

"Not so fast Abie, I can't let you do it..."

"Wait, it's not what you think. Look, it's the latest copy of OS/2 Monthly. We're carrying it here now." Boy, was I relieved. Abie and I settled down to a couple of egg creams and a good read. It's all a guy could ask for.

What does this all mean? The newsstand and subscription requests have sky-rocketed. And, in this issue, we are passing along the utility disk to all of our subscribers. All-in-all, OS/2 Monthly is becoming increasingly successful and we have all of you to thank.

I know you will find these utilities useful. I am a programmer by trade and constantly use most of what's on this disk.

If you are a developer and would like to submit a review copy of your software, send it, together with any promotional material you may have, to our address to the attention of Ron Beauchemin.

Regards for a healthy and happy New Year.

JOEL SIRAGHER

IMPORTANT NOTE:

Check out the OS/2 Technical Exchange February 28 through March 3. Call 800-438-6720 for further information.

EDITOR'S CORNER

With OS/2 IBM has proven that there is a demand for more powerful and elaborate personal computing. This revelation leaves Microsoft, which had established itself in the marketplace through marketing strategies, rather than marketable substance, playing catch-up with the ever increasing demand for a more formidable operating system. Microsoft is feverishly trying to present NT in this manner. Fortunately, the Microsoft marketing mavens hadn't anticipated the success of OS/2 and the remarkable comeback IBM is making in the personal computing arena. Microsoft without a monopoly on innovation, can no longer market on promises.

Windows was NT's precursor. OS/2 is Taligent's precursor. Assessing these two evolutionary waves, I feel more comfortable riding the latter.

With both operating systems, we will inevitably see a stronger foundation built upon object orientation and support for symmetrical multiprocessing. We will also witness the role the Mach kernel will play. Fruitlessly, we will try to count on one hand the number of other platforms upon which these systems exist. This and other magazines will devote innumerable pages explaining the possibilities laid out before us.

But, while we contemplate the future along with its possibilities, and place bets as to the progenitor of future operating systems, we tend to forget some of the other areas of OS/2 that contribute to the flexibility and desirability of OS/2 — foremost among them, REXX.

For some REXX — REstructured eXtended eXecutor — may seem a cumbersome language, not meritorious enough to be considered an integral part of OS/2, and certainly not worthy of its own column. But REXX's usefulness has eluded the general public. It was designed to be genuinely easy to use and effective at manipulating the symbolic objects generally used by people. The majority of publications about REXX, do not demonstrate its advantages over the existing DOS batch and OS/2 command file support. Learning REXX does not mean learning a new language, only expanding on what one already knows.

While REXX is not new to OS/2, as Eloy Cruz-Bizet points out in his column, it has been integrated into OS/2 with version 1.3 and was in the extended edition before that. Eloy instructs upon the usefulness of REXX and opens up new possibilities for beginners, which will surely increase their appreciation for OS/2. With his column, "REXX, The Modern Day King", novices are invited to "jump in" to REXX without spending a lot of time exploring its details. He explains how to replace crude DOS or OS/2 batch files with their REXX equivalents. This process of introduction provides the reader with a quick, simple, practical ice-breaker to REXX program-

ming. The reader is assumed to have only a limited understanding of the coding of simple batch files.

Skeptics who still believe that REXX is a waste of time, will be surprised when they see it pop up in a favorite application. Because REXX is essentially designed as a character manipulation language, it can exist as a macro language for up-and-coming applications. One such example is SourceLink. By implementing REXX as the macro language interface for SourceLink, we are not forced to learn yet another language. The same is done by CommandLine. A situation which could have demanded knowledge of the three languages (Command language, SourceLink and CommandLine) is reduced to knowledge of only one. It is a shame that more vendors do not provide REXX as their choice macro language. Those of you familiar with REXX will have no problem recognizing the name Steve Price. As a member of the IBM Endicott Programming Laboratory, he has contributed much to REXX's implementation, and offers his wisdom in this issue to broaden our understanding of REXX. For those of you who haven't guessed what OS/2's inherent answer to the phi-phenomenon is, it is REXX. "Discovering The Workplace Shell," takes a close look at how REXX integrates the Graphical User Interface with the Command Line Interface. With this article, the GMUI library is initiated and we encourage subscribers to contribute.

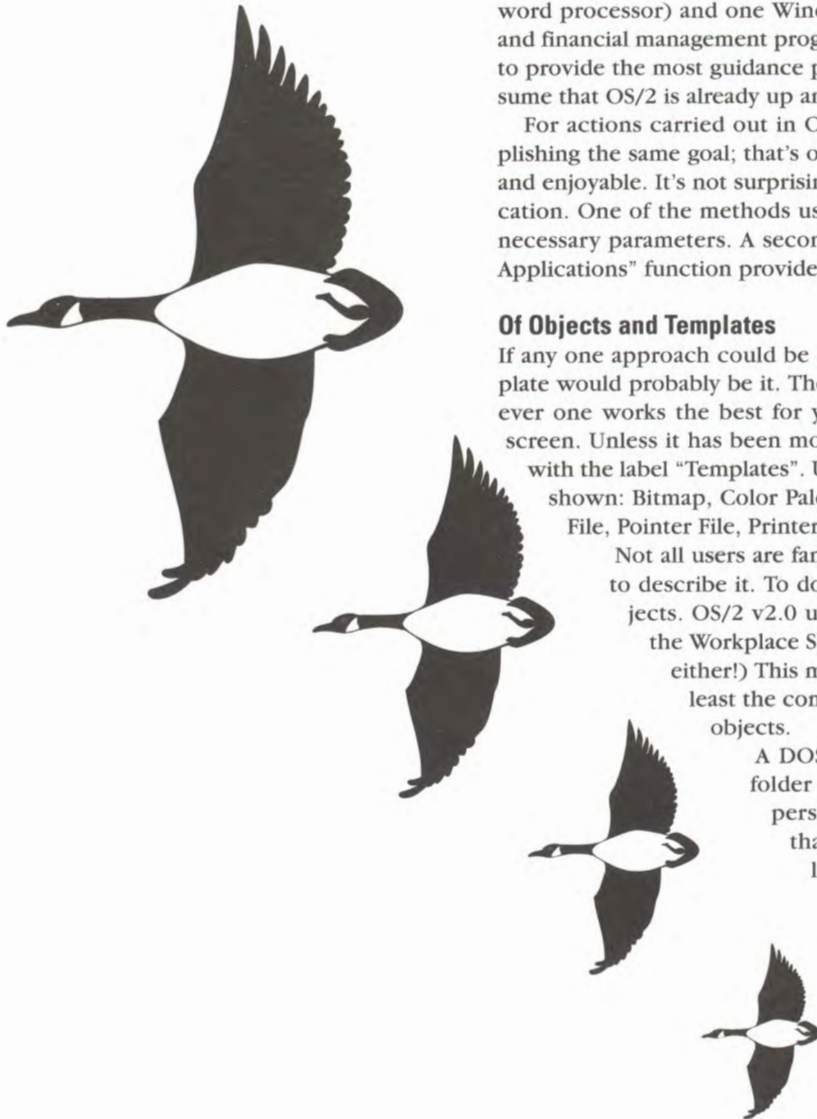
As more and more people migrate towards OS/2, our goal is to anticipate the needs of converts as well as veterans and to introduce them to some of the more obscure corners of OS/2. REXX is one of these areas. Whether the cover reads *OS/2 Monthly*, or (in a couple of years, perhaps?) *Taligent Monthly*, it is clear that IBM, with its brain-child OS/2, has made an indelible mark on the nature of computing and will continue to do so well into the future.

All the source code which is described in this magazine can be found on Peter Norliff's bulletin board. Look in OS/2 Monthly Section for the OS2 Jan.zip. The OS/2 Shareware BBS number is 703-385-4325.

BEGINNER'S SLOPE

Autumn Migration

**BILL
ZINSMEYER**



As a person considers upgrading to OS/2 v2.0, or after a person has (wisely) taken the plunge and installed it, several questions come to mind. Of course it can run most DOS, OS/2 and Windows applications, but how is an application defined to OS/2? Is it difficult? What steps must be taken? Do all existing applications need to be reinstalled from scratch, or can they be left on the hard drive and somehow introduced to OS/2?

Defining an application to OS/2 is called "Migration" (although the files involved do not move). This process, which is not difficult, can also be thought of as the creation of an OS/2 object to represent an application. Migration is the subject we'll examine this month, with some related tips thrown in as well.

This article will follow through the migration of one DOS program; WordPerfect* (a word processor) and one Windows program; Quicken** for Windows (a checkbook and financial management program). These applications will be installed from scratch to provide the most guidance possible for the greatest number of readers. We will assume that OS/2 is already up and running.

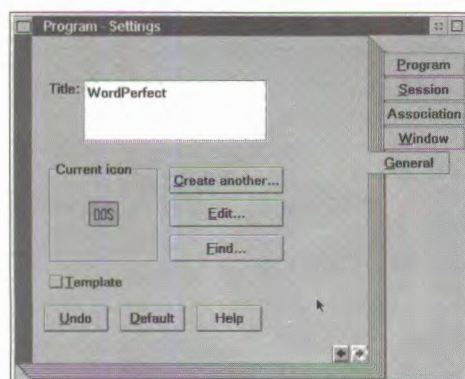
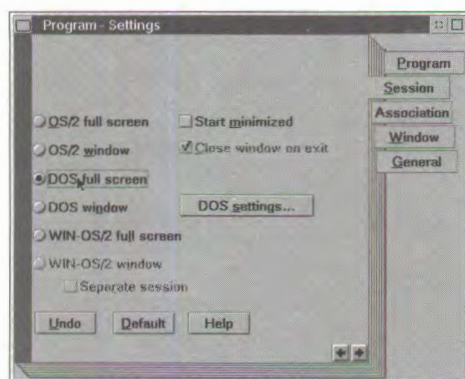
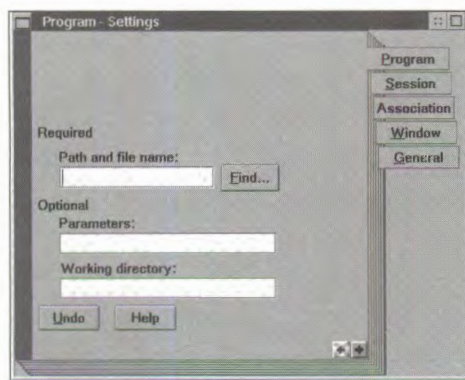
For actions carried out in OS/2, there are often several different ways of accomplishing the same goal; that's one of the features that makes the system so functional and enjoyable. It's not surprising then that there are several ways to migrate an application. One of the methods used quite often is to bring out a template and add the necessary parameters. A second method used frequently is to perform the "Migrate Applications" function provided with OS/2.

Of Objects and Templates

If any one approach could be described as the "official" one, then the use of a template would probably be it. The other methods are also valid, of course, so use whatever one works the best for you. To use a template, first go to the main desktop screen. Unless it has been moved to another location, a folder will be visible there with the label "Templates". Upon opening the folder the following 12 templates are shown: Bitmap, Color Palette, Data File, Folder, Font Palette, Icon, Metafile, PIF File, Pointer File, Printer, Program and Scheme Palette.

Not all users are familiar with the term "template," so let's take a moment to describe it. To do that, we must take a step back and also describe objects. OS/2 v2.0 uses an object-oriented graphical user interface called the Workplace Shell. (Hang in there, Beginners! I don't like buzz-words either!) This means that most of the components of the system, or at least the components that are visible on the screen, are considered objects.

A DOS window is an object, a floppy drive is an object, a folder is an object and a folder can contain other objects. A person familiar with Microsoft Windows might conclude that "object" is just another word for "icon". An object, like an icon, is a graphical representation of an element in the system (or perhaps of an interface external to the system) but objects have much greater functionality. In fact, an icon in OS/2 is just one small part of an object's makeup. The behavior of objects can be changed, associations can be established, different views can be accessed, they can be dragged and dropped, parameters can be specified and so on.



Peripheral devices such as printers are also represented as objects; even the main desktop screen is an object, although it does not have an icon.

A template is an aspiring object; it is an empty framework ready to be filled in and become a usable object. The variety of templates is provided, at least in part, to assist in migration. Rather than provide one generic template that would require extensive modification, several different types are available to simplify the process. As stated earlier 12 types are provided, but the three types most commonly used are Data File, Program and Folder.

WordPerfect

WordPerfect for DOS will be migrated using the template approach, but first it must be installed. From the main desktop screen, open up the folder labelled "OS/2 System", and from there open the "Command Prompts" folder. Double-click on "DOS Full Screen" to begin a DOS session. From this point the package can be installed exactly as it would be in native DOS, according to the instructions provided with WordPerfect. If utilizing the High Performance File System, do not install WordPerfect in a directory with a name longer than eight characters.

After installation is completed, migration is not essential before WordPerfect can be used. A DOS session can be requested just as mentioned above and WordPerfect can be invoked by moving to the correct directory and typing in "WP". Many advantages will accrue, however, if the application is migrated.

In preparation for migration, decide where the WordPerfect object will be located, open up the templates folder, and drag a "program" template to the new location. Dragging is accomplished as always in OS/2, by clicking and holding down the right mouse button while moving the mouse, except that when attempting to drag a template, you'll notice a surprising difference. The original template object, which can be thought of as a large stack of many program templates, will not move but instead will remain in the templates folder. What will be dragged is one program template that was "peeled off" the stack.

The behavior of templates (in the templates folder) is similar to that of another OS/2 feature users enjoy: the color palette used to change colors. When a choice is dragged out in order to change the color of something, the color also remains in its original place; it does not disappear. Templates resemble the color circles in this regard. The templates in the folder can be referred to as template stacks, to distinguish them from a template that has been moved out and is ready to be modified.

The next step to take after peeling off a template is to fill in the information necessary to make the template useable. This is done by altering the settings. "Settings" can be accessed through the pop-up menu, but will usually open up of its own accord the moment the program template is peeled off. If "Settings" does not open up automatically,

make certain it was a program template that was used. To access the Settings of any object, click the right mouse button on the object or template once. The pop-up menu that appears will have "Open" at the top, and next to the word Open will be a small box with an arrow in it. Carefully click once on this arrow box and the sub-menu that appears will have Settings as an option. Click once on Settings.

The feature now shown on the screen resembles a notebook and is sometimes called the notebook metaphor. The five "pages" of the notebook can be accessed by clicking on the tabs at the right: Program, Session, Association, Window and General. These pages differ depending on the type of object. As you become more confident with the Program notebook, you may experiment with settings not mentioned here. For now, only the minimum parameters necessary will be covered.

On the first page — Program — type in the name of the executable file for WordPerfect (WPEXE) on the first line. The directory in which WordPerfect was installed is probably not part of the path, so it will be necessary to type in the drive letter and the full path on the first line. Skip the second line (Parameters) and type in the drive and path for WordPerfect on the third line, leaving off the executable file.

On the second page — Session — choose the options DOS Full Screen and Close Window on Exit. WordPerfect for DOS works better under OS/2 as a full screen session.

On the next two pages, Association and Window, the default values are satisfactory. The defaults on the final page — General — are also acceptable, but here you will probably wish to change the name of the application to WordPerfect. Double click in the upper left hand corner to close the settings notebook.

Double clicking on the newly defined object (formerly a template) should now invoke WordPerfect. If it does not — if for example the settings notebook reappears instead — something was not entered correctly when specifying the parameters. Review the settings carefully. If WordPerfect does come up properly, congratulations! . . . You have successfully migrated an application to OS/2. This is much better than bringing up a DOS session and typing in the commands each time to get WordPerfect. For

AUTUMN MIGRATION

example, once a WordPerfect document has been created and saved, the object for the document can be dragged and dropped on the object for WordPerfect, and away you go! Remember that when using High Performance File System (HPFS), however, not only must the file names conform to DOS standards, but all directory names must as well.

Quicken

A common question asked with regard to Windows applications is whether the version of Windows that comes with OS/2 (WIN-OS/2) must be brought up on the screen every time an application is needed. The answer is no, not if the application is migrated. However, bringing up WIN-OS/2 is the proper way to install an application for the first time, so we will do so in order to install Quicken. In the "OS/2 System" folder, click on the object labelled Command Prompts, and then in that folder click on "WIN-OS/2 Full Screen". Depending on the PC, it may take a few minutes for the Windows screen to appear.

Once Windows is available, Quicken should be installed as per the instructions: choose "Run" from the Program Manager File menu. If installing from drive A, type A:\Install and so on, following the screen prompts to continue. As a result, Quicken will be installed, and just as WordPerfect did not have to be migrated to be used, the process could end here and Quicken could always be accessed by invoking WIN-OS/2. But like WordPerfect, leaving Quicken in this state would not take full advantage of OS/2's features, so we will continue on to migration.

Instead of using a template, in this case we'll use the "Migrate Applications" function that can be found under System Setup (which, in turn, is found in the OS/2 System folder). The window should say "Find Programs" at the top after requesting Migrate Applications. To save time and simplify the search, specify only the drive on which Quicken has been installed and select only "Windows Programs" in the box labeled Migrate.

After clicking the "Find..." button, Quicken should show up in a second window called "Migrate Programs". If it does not, select the "Add Programs" option, which will show a much longer, less selective list of programs. Assuming Quicken is available un-

GammaTech

Utilities for OS/2

Get the Best Performance and Reliability From Your Mission Critical OS/2 Work Station

The GammaTech Utilities are designed to enhance and complement your OS/2 system. Several utility functions are implemented in a Presentation Manager shell while others are provided in a command line mode for quick and easy access. These utilities provide the ability to perform the following maintenance and recovery operations easily without extensive technical knowledge:

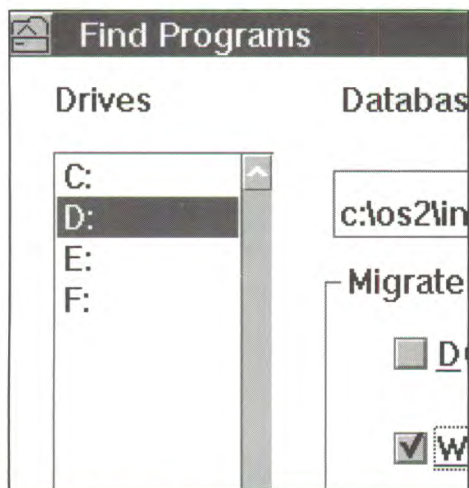
- Easy Graphical Installation
- SAA CUA Compliant PM Interface
- Optimize HPFS Volumes
- MS LAN Manager Support
- Recover Erased Files
- Obliterate Sensitive Data
- Mass Erase Selected Files
- Display Volume Information
- Alter File Attributes
- Add Comments to Files
- Alter File Time Stamps
- Locate Files
- Edit Disk Sectors
- Test Disk Media
- Display File Fragmentation
- Display Directory Information
- Supports HPFS386
- Supports Long File Names
- Supports Extended Attributes

The Best Time to Acquire Recovery Software is Before you Need it.

For OS/2 version 2.0 or 1.3. List \$125.00

VISA, MasterCard, COD, Purchase Orders
Federal Express Shipping for Fast Delivery

Post Office Box 70 Edmond, OK 73083
Phone (405) 359-1219 FAX (405) 359-7391



der Migrate Programs, select it, de-select any other programs that might be shown, and request Migrate. After a few seconds, a message should appear stating migration is complete. Exit from the Migrate Applications function. Quicken can be found in a specially-created folder labelled "Windows Programs" (or possibly "Additional Windows Programs"). You can examine the settings to learn what OS/2 has assigned, and change the settings if you wish.

Related Issues

Rather than opening up the Templates folder and choosing the appropriate template, an alternative method is to locate an object for a similar application and make a copy of it. The option "Create Another" can be found on the pop-up menu of most objects. The resulting object can be altered via the settings to invoke the new application; the settings can be customized in much the same fashion as is a template.

If there is a particular object used frequently to "Create Another", then perhaps that object should be given the ability to produce templates. To turn a normal object into a template-producing one, go into the settings notebook to the page labelled General. A box can be seen there simply marked "Template". Clicking the box on (giving it a check-mark) causes the object to behave like a template stack, while still retaining its original function as well. Not all objects have this potential.

During the course of this article, the reader might ask whether a template stack can ever be moved, given the fact that a normal attempt at a move results in a template peel-

ing off and the template stack remaining in place. The answer is yes, a template stack can be moved; just hold down the shift key while performing the normal move action. By the way, this is also the method used to move objects to and from floppy diskettes without the original remaining in place.

Most moves of graphical objects in OS/2 are "destructive moves". This term sounds dangerous but is not. A destructive move is one in which the object being moved appears in the new location and disappears from the old. In OS/2, if one wishes a copy-move to take place rather than the normal destructive move, the control key is held down in addition to using the mouse.

The designers of OS/2, after much research, decided it would be much safer to have OS/2 default to a copy-move rather than a destructive move when writing to or reading from a diskette. Thus even without holding down the control key, a move to or from a floppy will leave the original object in place. This can be something of a surprise because in so many other cases the original is lost, as expected. Remember, as is the case with template stacks, if one wishes to force the original to disappear (a destructive move) when going to or from a floppy, then hold down the shift key.

While on the subject, program objects cannot be moved or copied to diskettes, no matter what keys are held down. This is a minor drawback to OS/2 that IBM is considering addressing in the future. It's only a minor drawback because copying a Program object would not move the files for the application, it would just move a pointer to the application (plus settings, etc.), which would not have much meaning if placed on a diskette. At present, it seems there's already the potential to confuse File objects and Program objects, so perhaps the designers did not want to increase the confusion.

Here is something to be careful of: a small application, when viewed as a file on disk, can have exactly the same appearance as it would if the program had been migrated and had its own Program object. The File object looks as if it can invoke the application because it looks identical to an executable object. For example, the Jigsaw game in the games folder is made up of one file that looks just like the Jigsaw object when viewed graphically on disk. The difference is that the template underlying the

file is a File template, while the template underlying the executable object is a Program template.

If this sounds confusing, that's because it is, and some people say the OS/2 system should make the objects look different. To state the problem another way, you can click on the Jigsaw game all day and all night, but if what you're viewing is the File Object and not the Program Object, the game is never going to execute.

Although a Program object cannot be moved or copied to a diskette, it can be dragged to the shredder and deleted. Fortunately, the system is designed to delete only the object and not the file(s) associated with the object. This makes sense: to create a copy of a Program object does not create duplicates of the files involved; thus to delete the object should not erase the files. Dragging a File object to the shredder is quite a different matter, so take care: in that case the associated file will be deleted.

One final comment. This writer has not yet had the pleasure of installing a genuine, full-blown, 32-bit, Workplace-Shell-aware application designed specifically for OS/2 version 2.0. When I do, and if you do, migration should not be necessary at all. Migration is the creation of an OS/2 object for an application that probably wasn't written with OS/2 in mind, thereby providing a lot of OS/2 functionality the program did not originally have. A real OS/2 application will introduce itself to the system all on its own and of course create its own object. Several such applications are available now, and many more are soon to follow.

*WordPerfect is a trademark of the WordPerfect Corporation **Quicken is a trademark of the Intuit, Inc.

Bill Zinsmeyer is an Analyst and Programmer with many years' experience on mainframe computers. He greatly enjoys using OS/2 while learning more about PCs. Bill welcomes all comments, criticisms and, especially, suggestions for future topics — he can be reached via Compuserve at 71031,343

DISCOVERING THE WORKPLACE SHELL

REXX and the GUI

BRETT KOTCH

OS/2 2.0 expiates the lack of integration of graphic user interfaces (GUI) and command user interfaces (CUI) with REXXUTIL.DLL. REXXUTIL.DLL is a dynamic link library (a dynalink or DLL) that can be called from REXX and provides to the CUI user a means of accessing the GUI aspect of OS/2.

With dynamic linking, called functions do not have to be included as part of the executable file, but can reside in a DLL. A DLL makes it possible for the calling program to call routines external to itself as if they were part of the executable. These external routines usually may be written in any language and provide an interface to the calling program. In the case of REXX, these external routines must support the implementation-dependent interface used by the REXX language processor to invoke them.

REXXUTIL.DLL not only provides a means of integration with the graphical aspect of the WPS, but also with settings associated with the WPS. For example, you could start a DOS session with specific settings for that session, eliminating the need to have a special DOS shell object. With REXX and the REXXUTIL.DLL, you could search your entire hard drive and construct a folder which contains all of the reference books available. Or, probably the most requested of all, you could pop up a folder through the CUI of the directory you are in.

The following REXX code does just that.

```
/*obj.cmd*/
parse UPPER ARG shwid
call RxFuncAdd SysLoadFuncs, REXXUTIL, SysLoadFuncs
call SysLoadFuncs
if (shwid="" | shwid='.') then
    shwid=directory()
shwid=shwid';'
classname='WPShadow'
title=shwid
location='<WP_DESKTOP>'
setup='SHADOWID='shwid||',
      'OPEN=DEFAULT'

call charout, 'Building: 'title

result=SysCreateObject (classname, title, location, setup)

If result=1 Then call charout, '... Object created!'
Else call charout, '... Not created! Return
      code='result

Say";
return
```

The first object of interest is the assignment, `classname=WPShadow`. `Classname` is used to identify the

name of the object class: Run the following REXX script for a complete list of defined classes (Two other classes of particular interest are `WPFolder` and `WPProgram`).

```
/*REXX*/
call SysQueryClassList "list."
do i=1 to list.0
say 'Class 'i' is 'list.i
end
```

By using the class `WPShadow`, we are informing `SysCreateObject` that the object we wish to create is a shadow. Title, is just that, the title for the object that we will be creating. Location, determines where the object will be placed. In this case, the destination will be the desktop.

Other possible predefined locations are the startup folder: `<WP_START>`, the system folder: `<WP_OS2SYS>`, the templates folder: `<WP_TEMPS>`, the system setup folder: `<WP_CONFIG>`, the information folder: `<WP_INFO>`, the drives folder: `<WP_DRIVES>` and the hidden folder: `<WP_NOWHERE>`.

Up to this point, we have specified that we wish to create a shadow of an object and that the object should be placed on the desktop. Next, we go on to specify how we wish that object to behave with setup.

By setting the `SHADOWID` to `shwid`, the argument passed in through the command line, we are specifying the value that the object for which this object should be a shadow of. The value for this is a fully qualified pathname of a directory, program file or data file.

In this example, by just calling the script with no arguments, the script defaults to the present directory. `OPEN` defines how the shadow should appear once created. `DEFAULT` is the same as if the object had been double-clicked. Another possibility of interest for `OPEN` is `SETTINGS`: open settings notebook.

After these variables have been set — `classname`, `title`, `location`, and `setup` — a call to `SysCreateObject` will carry out our wishes for a folder representation of the supplied argument to `obj.cmd`.

While this is certainly a step in the right direction, it is certainly beyond the expertise of the beginner or novice. The information presented here came from various sources, most of which are only available with the developer's toolkit. What is necessary is a detailed specification for the new-found synthesis of the GUI and CUI and a greater realization from developers that they are truly not two different user interfaces, but one. ♦

FOR IMMEDIATE RELEASE

AirMouse Remote Controls

AirMouse Remote Controls is a hand-held, point and pick device that operates like a computer mouse, yet has no wires and requires no hard surface. The AirMouse works on a bidirectional flow of infrared light in conjunction with a tiny base station that sits near the monitor or projection systems. Its two-button simplicity, 10 meter (32 ft) range, and wide compatibility have enabled business and educational presenters, as well as trainers, world-wide to bring simplicity and ease back to their presentations. OS/2 drivers are available.

Selectech Ltd, 30 Mountainview Dr, Colchester VT 05446, 802-655-9600

BRIEF 3.1

BRIEF 3.1, professional programmer's editor is now being shipped by Borland. They offer both the DOS and OS/2 version in one package. The previous standard features Macro Language and Debugger, Language compilation within BRIEF, Multi-language support, Smart indenting, Statement completion and Multiple window support remain a flagship of the product. Borland has introduced Mouse support, Extended Memory, Redo and Dialog Box Support.

Borland International, 1800 Green Hills RD, PO Box 660001 Scotts Valley, CA 95067-0001, 408-439-4833

32-Bit Relish and Relish Net

Sundial Systems released a 32-bit version of Relish, and Relish Net. The basic client servers design of the Relish scheduling products exploits OS/2's multi-tasking and multi-threading capabilities. Relish provides intuitive and reliable personal calendaring in the OS/2 environment. Relish Net provides transparent real time access to the schedules of the LAN-base workgroup; it is the only Presentation Manager group scheduling solution. Relish 32-bit and Relish Net 32-bit provide extensive drag-and-drop support, expanding the applications' scheduling/rescheduling flexibility to take full advantage of the Workplace Shell.

Sundial Systems, 909 Electric Av, #204, Seal Beach, CA 90740, 310-596-5121

EASEL Transaction Server (ETS) Toolkit

Easel Corp has begun shipping the EASEL Transaction Server (ETS) Toolkit and the EAD/SQL Option for the EASEL Workbench, two new products that provide developers with powerful features for building client/server applications. ETS Toolkit provides developers with high-speed static SQL access to leading client /server databases. It works in conjunction with the company's EASEL Workbench Development Environment to enable developers to build online transaction-processing applications which require high-speed processing of complex database transactions.

EASEL Corp, 25 Corporate Dr, Burlington MA 01803, 617-221-2100

SAROS Mezzanine Network Engine

SAROS Mezzanine System provides the best available solutions to file management, online archiving and complex administration problems in large multiple-server networks. The powerful Saros Mezzanine network engine allows you to offer the benefits of low-cost PC client-servers computing to large network users. Plus, it gives you the flexibility to integrate your choice of single-user applications through one of the several user interfaces offered by their developers guild.

SAROS Corporation, 10900 N.E. 8 St, 700 Plaza Center Building, Bellevue WA 98004, 206-462-0879

OPAMP Technical Bookstore

OPAMP Technical Books has been recongnized as the leading source for books on OS/2, as well as all aspects of computer science. Call or write for their free 300-plus page catalog.

OPAMP Technical Bookstore, 1033 N. Sycamore AV, Los Angeles, CA 90038, 800-468-4322

Personal REXX and REXXLIB

Quercus Systems, a leading third-party developer of REXX language-related software, has announced that on December 1, it will begin shipping a 32-bit version for OS/2 2.0 of its principal product, Personal REXX, and a new product, REXXLIB, which is a 32-bit enhancement library for IBM's OS/2 REXX. Personal REXX is Quercus Systems' implementation of REXX, the SAA standard procedure language for OS/2 and DOS. It is compatible with IBM's OS/2 implementation of REXX, but offers superior functionality, performance, documentation, and support. Personal REXX is the most mature implementation of REXX for the PC platform, having been available since 1985.

A significant advantage of Personal REXX over IBM's REXX is its extensive set of added library functions. Most of these are also being made available separately in the new REXXLIB product. This is a toolkit of essential functions that gives full access to the advanced capabilities of OS/2 and makes possible rapid development and prototyping of applications which use REXX.

Many of the functions enhance REXX support for compound variable "arrays". Functions are provided for sorting arrays, finding all "tails" of a compound variable, copying and combining arrays and reading or writing array contents on disk.

Another class of functions adds many important mathematical routines to REXX. This support includes trigonometric, logarithmic, and exponential functions.

A third category of functions provides extensive text-mode screen support for developing windows, menus and data entry panels in REXX.

Many additional miscellaneous functions provide a variety of system services, such as date-format conversion, listing of file-extended attributes, file system services, listing of active system processes, and support of interprocess communication.

FOR IMMEDIATE RELEASE

For a long time, serious users of the REXX language have requested, without success, that IBM provide many of these capabilities — such as the mathematical and array-handling functions.

Personal REXX will continue to offer serious REXX users a superior language package in various areas beyond the scope of REXXLIB. For instance, it includes a "global variable" facility, similar in concept to system environment variables, but more general in nature. This facility supports true system-wide global variables that can be used for communication between independent REXX programs running in any process within the system. Personal REXX also relaxes certain limitations of the IBM REXX implementation, such as the number of arguments which can be passed to a subroutine, the allowable depth of subroutine recursion and the size of the largest program which can be saved in a "tokenized" form for best performance.

In spite of these enhancements, Personal REXX is a compatible extension of IBM's REXX both in the language itself and in the application programming interface, so it can be used as a transparent plug-in replacement.

Personal REXX and REXXLIB are available directly from Quercus Systems and from dealers. The retail price of Personal REXX for OS/2 2.0 is \$175, and REXXLIB is \$50.

CORRECTION:

SQA:Replay was created by Software Quality Automation, 1 Parker St., Lawrence, MA 01843.

REXX is the key that unlocks the full potential of OS/2

PERSONAL REXX™ is the complete REXX package for OS/2

If you're just getting into REXX, **Personal REXX** offers:

- A full 32-bit implementation of REXX that is the fastest available
- Complete documentation in the package
- Free support by telephone, BBS, and CompuServe
- Seven years of experience with REXX on a desktop platform

But if you are already a REXX expert, there's much more:

- True system-wide global variables
- Interprocess communication support for named pipes, semaphores, and the REXX external data queue
- Array operations like copying, sorting, and searching
- Advanced mathematical functions (trigonometric, logarithmic, exponential)
- Text-mode user interface support for windows, menus, and data entry screens.
- Date format conversion, regular expression file searching, and many other system services
- Full compatibility with IBM's OS/2 and mainframe REXX implementations

If you prefer, most functional enhancements are also available separately in our **REXXLIB™** add-on function package for OS/2 REXX.

Personal REXX is also available for DOS and Windows.

**QUERCUS
SYSTEMS**

P.O. Box 2157
Saratoga, CA 95070
Phone: 408-867-REXX
Fax: 408-867-7489
BBS: 408-867-7488

How much time do you waste searching for bugs?

Toolkit API calls usually work. So that you don't have to check every return code, Error Manager does. EM tells you which API failed, where it failed and why. EM can log all errors and your messages to a file, pipe or to our viewer. And, it optionally notifies your window procedure enabling centralized error processing.

Error Manager 32-bit Version 2.0 for OS/2 2.X **\$225***

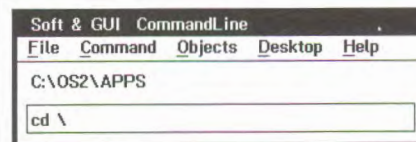
- Invaluable for Debug, QA Test and Production cycles.
- Finds intermittent errors without the Debug Kernel.
- Works with optimized code in realtime situations.

Session started on 11/2/92 at 4:11:14
Programmer Message - This is my message 1
WinGetPS() generated error 1001 - Invalid window handle in file TEST.C at line 19.
DosRead() generated error 6 - Invalid handle in file TEST.C at line 21.
Programmer Message - This is my message 2

Include 1 header file.
Link 1 DLL.
Set 1 environment variable.

Includes a PM Viewer,
Pointer Validity API and a
Free copy of Command Line

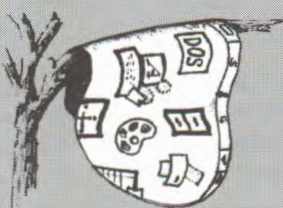
How much searching for files, folders, objects and sessions?



32-bit Version 2.0 for OS/2 2.0

Command Line™ \$39⁹⁵* Quit folder diving! Launch any object instantly!

- Intelligent Application Launching with:
- Hotkey Application Start & Switch
 - Command Aliasing & Persistent History
 - Inline File Find & Filename Completion
 - Alternate Command Shell Support
 - REXX Expression Interpretation and Calculation
 - Instant Workplace Shell Object Creation
 - Desktop Cleanup & Quick System Shutdown
 - Ideal for Laptops, Portables & Vertical environments

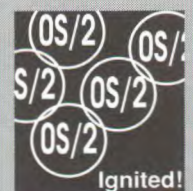


Soft & GUI

2224 East 21st Street
Brooklyn, New York 11229

(718) 769-8017

* NY State residents please add sales tax.



REVIEW

- **HyperAccess/5**
- **Parts Workbench for OS/2**
- **COMDEX**

**RON
BEAUCHEMIN
&
ERIC
PINNELL**

HyperAccess/5

OS/2 character-mode application

The HyperAccess/5 (HA/5) communications software from Hilgraeve, Inc. is available in 3 different packages which all use a common communications engine. There is a DOS-only package, a Windows-only package and an OS/2 package which contains both the DOS and OS/2 versions. The OS/2 package will be the subject of this review.

HyperAccess/5 version 2.1 is an OS/2 character-mode application (a PM version is due out before the end of the year and will be covered in a later review in OS/2 Monthly). The manual provided with the software contains 325 pages plus an index. The manual is well organized and easy to use. Novice users should not be intimidated by the size of the manual, since over 150 pages are dedicated to HyperPilot, the scripting language for HA/5.

The installation procedure is fully automated and straight forward. HA/5 comes with setups for over 160 modems, including higher-end modems such as the U S Robotics Courier V.32bis modem used for this review. This saves the user from having to deal with cryptic modem initialization strings.

An automated script-generation facility is provided so that a HyperPilot script can be generated by memorizing the user's actions. This feature is referred to as the "learn" mode in most communications software that provide that feature. Accordingly, the user invokes the learn function by depressing Alt-L and ends the learn session by depressing Alt-L. I tried this feature with a few different bulletin boards and it worked well.

Hilgraeve uses the dialing directory to store entries for each service that the user wishes to call. The dialing directory is displayed when HA/5 is initiated along with a one-line mini menu at the bottom of the screen. The mini menu shows the key combinations needed to access major categories, such as communications and learn. Once the user depresses one of the key combinations from the mini menu, he is guided through every selection necessary to accomplish his objective. This design allows

the program to be used with little or no reference to the manual.

A point and shoot file selection capability is provided. The user can access this function at any point where a file name is required by simply depressing Alt-D, as indicated on the mini menu at the bottom of the screen. The point and shoot facility allows multiple file selection in a single pass for protocols such as Hyper-Protocol, ZMODEM, YMODEM and YMODEM G that support multiple files in a single upload. The multiple-selection option is invoked by depressing Ctrl-Enter on the first and last files of the group to be selected. Files in the middle of the group are selected by pressing enter.

The editor provided with the software allows up to two windows. Cut and paste capability is provided between windows. The editor can also be used to mark a block of text that can be sent without sending the rest of the file. Users of remote interactive systems might find this feature helpful. Search and replace capability is also provided within the editor. Those who wish to use their favorite editor instead of the HA/5 editor may do so by simply updating a parameter within HA/5. The parameter contains the commands necessary to invoke the program from the OS/2 command line. Multiple commands are allowed. The user simply separates the OS/2 commands with a semi-colon. The following string might be used to invoke the OS/2 Enhanced Editor: `CD\OS2\APPS;EPM.`

The answer mode within HA/5 provides more than just the usual auto-answer capability. In fact, HA/5 provides a lot of the capabilities of remote control programs, such as Carbon Copy and Remote2. This facility allows the user to start up HA/5 as a background task to provide remote access to his PC while he is doing other work. That other work could even be setting up another remote access task, providing a second comm port and modem are available on the machine.

Remote access capability is great, but without proper security it would be dangerous at a minimum and probably forbidden by the Information Services staff, the auditors or both.

**See your entire system
graphically at a
glance.**

**New OS/2 Tree utility is a super
Programmer, LAN Administrator
Productivity Tool**

- Complete programmability every function/key can be customized!
- Display your tree structure for all system and LAN disks concurrently.
- Manipulate your files, directories, entire disks and/or entire system!
- HPFS fully supported
- Pop-up lists of files for any directory
- Edit/browse file(s) using any EDITOR!
- Custom task manager
- Custom document formatting and printing
- Multiple pop-up to-do lists
- Date and Appointment reminder
- Display system clock in any application
- Upload/download files between mainframe and PC
- Custom programming language for macros and batch operations
- 30 day unconditional guarantee.

**Intro Price: \$49.99
\$249.99 Professional Version
Network Licenses Available**

**Levine Computer
Consulting Services
2501 Porter St.
Suite #838
Washington, DC 20008**

**Orders / Fax / Inquiries / Helpline
(202) 966-5309**

REVIEW

Hilgraeve has done a good job of addressing the security concerns. There are several levels of security within HA/5. The first level is password validation. The second level is auto-callback. In addition to associating a password with the remote user's name, the user also supplies the phone number of the remote user's modem. After the remote user supplies his valid password, the system will automatically call the remote user back at the telephone number associated with the remote user's name. The third level of security centers around access rights. The access rights are read, write, overwrite, print, file management, drive and/or directory restriction, OS/2 command line access and program/script restrictions. In addition to the security controls mentioned above, HA/5 provides the ability to log the date, time, name of the caller, session duration and commands issued by each and every caller.

The file transfer protocols supported by HA/5 are HyperProtocol, ZMODEM, YMODEM, YMODEM G, KERMIT, XMODEM, 1K XMODEM, Compuserve Quick B and straight ASCII. All those mentioned above are very common, with the exception of HyperProtocol. HyperProtocol is a Hilgraeve protocol which does on-the-fly compression and decompression. If both parties have the HyperProtocol capability, it should be used. HyperProtocol will be the fastest and most efficient at any baud rate. A good comparison of the supported protocols is provided in the manual. This comparison also highlights common misconceptions about the various protocols.

HA/5 is the only communications program I know of that provides on-the-fly virus protection. HA/5 uses virus signatures provided to them by the IBM virus research group. If HA/5 detects a virus within the file being received, it immediately displays a message identifying the virus that was detected. The user may then abort the transfer or continue to receive the file. If the user does not respond to the message within 30 seconds, HA/5 assumes the user is not at the keyboard and aborts the transfer, after logging a message to that effect in the call log file.

HyperPilot is the scripting language provided with HA/5. HyperPilot's scripting capabilities, while not as extensive as the Crosstalk Mark 4 CASL scripting language,

are more than adequate for even the most demanding application, when combined with the multi-threaded capabilities of OS/2 2.0. The structure of the script commands is very similar to function calls in the "C" programming language. The commands and parameter names are easily-understood English words and are well documented. Hilgraeve dedicates over 150 pages of the manual to the HyperPilot scripting language.

There was only one inconsistency that I noted in the whole package. The DOS version of the program that is bundled with the OS/2 version does not support IRQ levels above 7. The OS/2 version does not experience this problem, since it just writes to the name of the comm port and OS/2 handles the rest. OS/2 2.0 natively supports IRQ levels up to 15 via a parameter on the COM.SYS device line in the OS/2 2.0 config.sys. Most users will not care about the IRQ situation with the DOS version, since it will only affect users who have a 16-bit serial card that supports IRQ levels above 7.

HyperAccess/5, version 3.0, is currently in the late stages of beta. Version 3.0 is also a character mode application that contains several enhancements to the 2.x version of the product. Those enhancements include mouse support, on-the-fly decompression of .ZIP files as they are being downloaded, virus protection for .ZIP files, an integrated spell checker, support for over 200 modems, and various file transfer speed and recovery enhancements. Current HA/5 users can upgrade for \$29.95 plus \$6.00 for shipping and handling.

HyperAccess/5 is a superior communications package that comes with a 60-day money-back satisfaction guarantee. This is as close to shareware as you can get with commercial software. I would highly recommend that anyone considering the purchase of an OS/2 or DOS communications package give HyperAccess/5 very serious consideration.—RB

The OS/2 & DOS combination package is priced at \$199, while the DOS only package costs \$99. Hilgraeve can be reached at 800-826-2760 or 313-243-0576. The FAX number is 313-243-0645.

Parts Workbench for OS/2

Object-oriented programming tool

Last August, I attended the Windows and OS/2 conference in Boston. An IBMer at IBM's test-drive center showed me a software tool called PARTS (Parts Assembly and Reuse Tool Set). It's not often I get excited about software, but one look and I was hooked. It's made by Digitalk, the same folks who make Smalltalk V.

PARTS is not your typical GUI builder, but an object-oriented programming tool that uses a variety of objects (a.k.a. parts) to perform sophisticated functions for the user. There's no need to write any code, only to click on an object and drag it to the appropriate position on the screen. It's that easy.

Users can also link the objects to other objects or system variables. For example, it is possible to create a dial gauge that is linked to a slider. As the slider is moved, the indicator on the gauge changes.

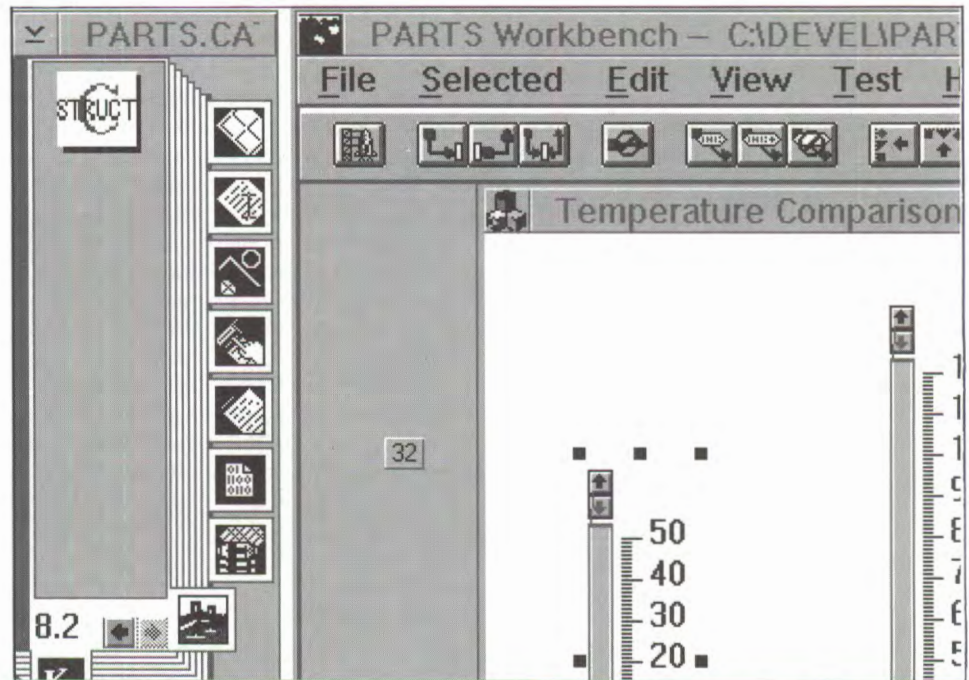
There are many different types of objects that come with PARTS. There are the usual assortments of buttons, sliders, plus some unusual objects like speakers, graph panes, timer, printer, various data entry fields etc.

In addition, new customized parts can be created by the use of a powerful built-in scripting language — writing modem parts, graphics parts etc.

PARTS is designed to be linked with other programs and objects as need be. By using an accessor, it is possible to add PARTS functionality to your existing C programs. There are many types of accessors in PARTS, such as the Printer, Speaker, Launch pad, Clipboard, DDE, DLL, Timer, File Disk and Btrieve accessor. These accessors give PARTS the capability to be integrated into a wide variety of different areas.

Parts can be used to create stand-alone applications. Users are allowed to freely distribute the runtime DLLs for PARTS (roughly about 1 megabyte in total disk space) without royalties. The .EXE files created by PARTS are quite small, so your application shouldn't be much over 1 megabyte in size.

There are, of course, certain down sides to using PARTS. Firstly, developing applications in 8 megabytes of RAM on a 386-25 is a slow process. More RAM would clearly help avoid excessive paging. Secondly,



PARTS is not as cheap as other tools (like Borland's ObjectVision) with its \$1995 price tag. Fortunately, Digitalk offers a 60-day money-back guarantee, so users can see if the price tag is justified. Finally, PARTS is currently available only for OS/2, and it is a 16-bit version. PARTS does not act as a code generator, which some developers may require. On the other hand, PARTS offers one of the most powerful interface-building systems in existence, and its ease of use is without par.

I spoke at length with Steve Dwyer from Digitalk about future plans for PARTS. Digitalk is working on a 32-bit version, to be released in December. This version will be made available free to all current PARTS users. It will have some bug fixes (although, I must confess, for a 1.0 release, PARTS is extremely stable) plus a bunch of new parts. This new version will also include come CUA91 controls and containers.

Digitalk is also working on a Windows version, tentatively due in the middle of 1993. When I asked Steve how easy it would be to port an application from the OS/2 version to the Windows version, and he assured me that code would be highly portable. I also asked about a Macintosh version, and it seems likely that we will see PARTS for the Mac (although I wasn't given a time frame).

PARTS is extremely powerful in its capabilities, and for my own personal consideration, it's worth the \$1995 price tag. It's easy to use, stable and offers more fun creating applications than I have ever had. However, if you need cross-platform capabilities, PARTS is not for you. — EP

PARTS is available from Digitalk Inc., at a suggested price of \$1995. They can be reached at (310)-645-1082.

COMDEX

OS/2 Hits a Home Run at Fall Comdex

CRACK!! That was the sound of OS/2 hitting a home run at Fall COMDEX. Our favorite operating system won THREE awards: PC Computings' MVP award (for OS/Environment), PC Mags' award for Technical Excellence (Best OS), and PC Worlds' "Most Promising Newcomer." This sort of praise from major computer magazines was, to say the least, somewhat unexpected. Perhaps now we will see OS/2 get the media recognition it deserves.

I decided that in my foray into the wilds of the COMDEX trade show, I would try to dig up answers to the two most important

REVIEW

questions: where are all the SVGA drivers and where are all the 32-bit OS/2 appls?

The video driver situation is getting better. S3 reports that it will have drivers ready for its chip set as of mid-December. However, these are not seamless-drivers (those should arrive at the end of March). Weitek is busily working on driver support for its family of products, and should be ready some time in 1993. Western Digital is already supplying drivers to IBM, but seamless support will be "early in 1993". In other words, most major chip makers will have drivers by the end of 1993. Unfortunately, ATI Technologies is one of the slower ones, and is promising drivers for its Mach 8 and Mach 32 chip set in mid-1993. Rumors of the IBM beta of BORG (a.k.a. OS/2 s.1) contradict this, claiming drivers for many major boards will be included. CD-ROM drivers for several manufacturers are already appearing in IBM beta code and I expect that by the time OS/2 2.1 ships, (1993) most users will have proper CD-ROM support.

On the apps front, several major ISVs

were demonstrating their wares. Lotus was demonstrating Ami Pro, Lotus 1-2-3, and cc:mail for OS/2. These products use threading and object-oriented interfaces wherever possible. My favorite, however, was cc:mail. Not only is this product faster and easier to use than the Windows version, but Lotus has added extensions to the REXX programming language so you can automatically have your computer get your mail for you at preset times. CC:mail is due in February. Most of the other Smart Suite apps are due late in 1993. Also look for Lotus Freelance.

Computer Associates is porting over several of its applications (Realizer, Compete, Unicenter, etc.) over to the OS/2 environment. They, too, are embracing the object-oriented paradigm. Interestingly enough, CA plans to market its Windows and OS/2 versions in the same box. These applications should start appearing early in the second quarter of 1993.

CA was also showing off its Unicenter LAN management tools. This utility lets you schedule jobs, do problem reports, perform

system security and so on. Unicenter supports both DSM and CID. I was struck by how easy it was to configure the system (maybe there is something to this object-oriented approach after all, eh?). According to CA, some 20% of their clients have decided to standardize on OS/2, which is a compelling reason CA decided to port its apps over to OS/2.

The smaller ISVs were busy too, IBM OS/2 ISV area had roughly 100 exhibitors. Popular products included Norton Commander for OS/2 (now shipping), as well as demos for Ami Pro, Lotus 1-2-3, DeScribe 4.0 (now shipping). Hilgraeve plans to ship its PM version of its HyperAccess communications software in 1993. Artisoft has plans for OS/2 version of its LANtastic networking software. Expect to see it in 1993.

IBM has finally reduced the price of its OS/2 PL/1 programming language to \$495 following pressure from its customers. PL/1 users should be pleased with the result. LAN Server 3.0 is now shipping, and IBM claims it to be the fastest network operating

Tennis Pros, Golf Pros And OS/2 PM Programmers Have One Thing In Common...

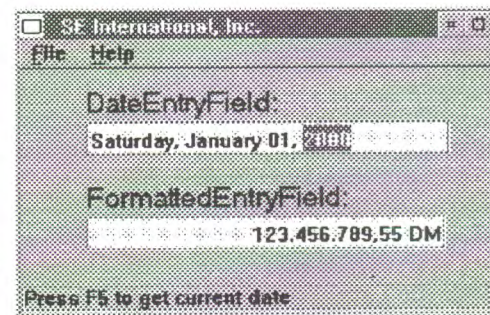
...they need proper training and tools to be successful !

The right mix makes all the difference! PM programming is easier than you think **IF** you use proper object-oriented techniques and high level PM window classes.

Ask for **SE International's**

- ✓ **Inhouse PM programming training.** Within one week you learn how to succeed in large C and PM projects and how to choose the right development tools.
- ✓ **PM Extensions.** Powerful new 16-bit and 32-bit PM window classes (**FormattedEntryFields**, **DateEntryFields**, **PrimaryWindowClass**, **Animated Bitmaps...**) allow you to focus on the application logic and not on programming of frequently required window behaviors. All our **PM Extensions** are **Gpf™** compatible.

SE International, Inc.
621 NW 53rd Street, Boca Raton, FL 33487
Tel: (407) 241-3428 • Fax: (407) 997-8945



SES Software Engineering Services GmbH Berlin
Koenigsallee 49, D-1000 Berlin 33, Germany
Tel: +49 30 826 40 63 • Fax: +49 30 825 30 14

™Gpf is trademark of Gpf Systems, Inc.

system, bar none. This speed is a result of having 32-bit HPSFS code. Support for Macintosh clients is included in LS 3.0.

One product that caught my attention was the Multimaster multimedia database by Commix. This product allows you to combine still images, digital video, and database programs all at once. So, by clicking on parts of an image, you can have it run various clips. The sample program was a walk-through program for real estate agents. So, by clicking on a particular room, you could get digital video and/or sound of the agent describing/showing that room. This product is scheduled to ship December 15, at a suggested retail price of \$195.

Of course, no visit to COMDEX would be complete without a visit to the IBM booths. I was stunned to see a working version of the IBM 486SLC3 clock-TRIPLING CPU. That's right folks, 33 MHz in, 99MHz out. Sampling of this chip will be in 1993, with production starting in 1993. Interestingly enough, since the chip is based on a lower (3.6) voltage circuitry, it only consumes about 25% more power than a typical DX2. Other neat hardware included the ThinkPad 700C notebook computer. Its active matrix color screen produces vivid colors, although I must confess that I do not like its little "eraser" mouse built into the keyboard.

So what other fancy hardware does IBM have planned? Well, for starters, it is building a new video board that will incorporate a special socket into which users can insert an ASIC chip. This chip will be used to compress and decompress digital video on the fly, thus relieving the strain off the CPU. This board may well become the XGA-3 board. Those of you who do not need multimedia, don't bother buying the add-on.

The hit of the show, however, was seeing OS/2 running on the Mach microkernel(!). IBM has decided to create a new OS called the Workplace OS. This is essentially a Mach microkernel with a series of add-on "personality modules". There will be modules for DOS, Windows, OS/2, and AIX. And the good news is you can expect to see this technology begin beta testing by the end of 1993. The Workplace OS is, of course, IBM's answer to NT.

IBM plans to introduce a new version of DOS that has a Workplace Shell interface. Called Workplace DOS, it will support both 16b-bit and 32-bit applications. This product

is designed to run on low-end systems and compete directly with Windows. You can expect this product to include both a disk compression program (Stacker) and memory management utility (386 Max). Applications will probably be cooperatively multitasked a la Windows. You can expect to see this DOS as a ROM-based product, which is designed to compete directly with Windows on the low-end machines.

OS/2 is also to be enhanced. The first major upgrade is OS/2 2.1 (code-named "BORG"). This release will feature Windows 3.1 support, including the provision for running enhanced mode Windows apps that require the WINMEM32 DLL, and TrueType support (for Win Apps only). Expect to see Pentium and 486 specific versions of the operating kernel included. Borg will also feature power management utilities (expect a 25% increase in portable's battery life), as well as PCMCIA support. There will be increased printer, video, and CD-ROM support. Expect to see this product late in 1993 (IBM is finally taking the issue of serious beta testing to heart).

On the Multimedia front, IBM now has its Ultimotion digital video running at 640x480 resolution (12 frames per second) on a standard XGA-2 card. Perhaps there is hope for this technology for the masses without requiring fancy hardware. Speaking of fancy hardware, IBM (in conjunction with TI and Archer Electronics) is working on a special digital sound board that will support the M-Wave Digital Signal Processor, and Archer's Q-Sound 3D Sound system. This will allow for realistic 3D sounds to be projected from speakers and will include advanced features. Expect to see the product in 1993.

Speech recognition was being demonstrated by IBM using its RS/6000 server on a LAN. Using highly sophisticated analysis techniques, the speech recognition rate was highly accurate. I suggested, however, that this product would be better suited for Intel's Pentium chip, primarily because Pentium will have a higher market penetration than RS/6000. Let us hope IBM listens.

According to Lee Reiswig, as of fall COMDEX there were over 600 32-bit OS/2 apps shipping, with an additional 75-100 applications arriving per week. This is good news for the OS/2 community, which has been starving for 32-bit OS/2 apps. However, the big push in terms of applications

availability will be in 1993, as the big names like Lotus and Computer Associates unveil their wares.

At a recent seminar hosted by Mike Kogan in Toronto, some of IBM's plans for OS/2 were revealed. Future enhancements due by the end of 1993 include the ability to have symmetric multiprocessing (SMP) on X86 systems. There will also be integrated networking (a la WFW). Support for DCE is promised. You will also see security and disk encryption available, making OS/2 C2 compliant. OS/2 will receive asynchronous input queues, thus curing it of a potential weak spot. Also expect to see 32-bit PMWIN code, as well as PM OLE (Object Linking and Embedding), Distributed PM (Similar to X Windows), Win32s support, double-byte character support, and support for Japan's popular DOS/V. —EP

INDEX OF ADVERTISERS

Adaptive Research & Design	22
Arcadia Technologies, Inc.	A-3
Bon Ami	24
Ceres Technologies, Inc.	26
Clear & Simple, Inc.	36
GammaTech	6
Gpf Systems, Inc.	Cov 4
Guild Products Inc.	A-4
Hierarchical Applications Ltd.	23
Hilbert Computing	37
Information Builders, Inc.	A-1
JP Software	34
Levine Computer Consulting	12
Lotus Development Corp.	Cov 3
Mitnor Software	25
MSR Development	18
Oberon Software	47
Ontos, Inc.	33
Orange Hill Software	36
Peter D. Norberg Consulting	45
Quercus	10
RimSTAR Technology	28
SE International, Inc.	14
Soft & GUI	10
Softbridge, Inc.	35
Softnet	A-2
The Stirling Group	Cov 2
Sundial Systems Corp.	30-31
ViaGraphics	27

REXX: A MODERN DAY KING

Comments, Labels, and System Commands

Abstract

Novices are invited to "jump right in" without spending lots of time exploring the details of REXX, a powerful and function-rich procedure language intended to replace DOS-like batch files in OS/2. The REXX procedures language was developed primarily with the beginner in mind and has retained ease of use as its most valuable function. Replacing crude DOS or OS/2 batch files with REXX equivalents provides the reader a quick, simple and yet practical ice breaker into elementary REXX programming. The reader is only assumed to have a limited understanding of the coding of simple batch files.

Why REXX? Ask any well-versed programmer about the benefits of using the REXX language as the tool of his craft and you'll get a variety of answers. Some tell of the string manipulation facilities or about being able to use variables without declaring them, while others rave about the built-in functions. Because REXX is simple to learn, flexible, and highly forgiving, how easy it is to start programming is always among the responses — not to say that an expert couldn't do wonders. REXX is a language designed to attract pilgrims. The primary goal in the architecture of REXX, says Mike F. Cowlshaw, its designer, is "that it should be genuinely easy to use both by computer professionals and by casual users, in the belief that the best way to foster high-quality programs is to make writing them as simple and enjoyable as possible." With the information and examples in this series of articles, the reader can participate in this enjoyment, learning fundamental elements required to translate a DOS or OS/2 batch file to the ease-of-use world of REXX. Each installment of this series allows the reader to approach increasingly more difficult types of batch file translations. The length of this article does not permit exhaustive review of each REXX keyword. The bibliography refers the reader to notable references.

Experiment! It enhances any learning experience, but making changes to any of your existing "live" batch files without taking a viable backup is like flying without a parachute.

In this article, we deal specifically with comment statements, program labels, simple unconditional branching, simple quoted strings and, finally, system commands. You'll be taught to convert a simple batch file containing just these elements to REXX. Table 1 shows keywords of batch files for OS/2.

REXX in OS/2

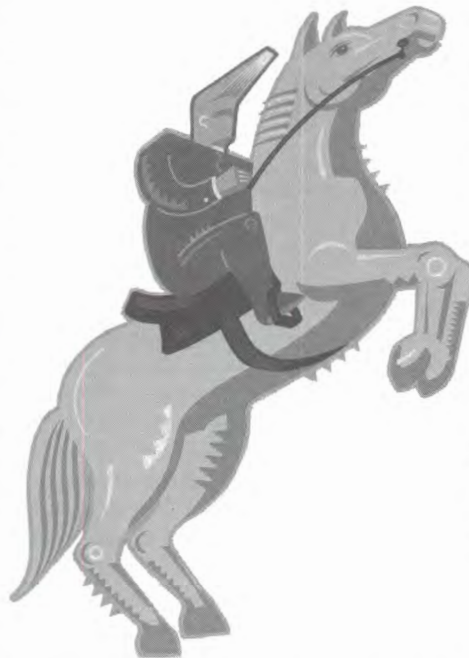
Since version 1.3 of OS/2, REXX has been integrated into the Standard Edition it is IBM's intent to allow both batch files and REXX programs to coexist. To the OS/2 file system, batch files and REXX procedures share a common "filetype" suffix ".CMD". (Note the

difference from the DOS ".BAT"). Both types are processed by the command interpreter, and before actually read, not much is known about their content. This implies that a directory containing ".CMD" files may include both types without conflict. Upon executing a ".CMD" file, the file is opened and read. If the interpreter finds the characters "/" in the first two columns of the file, it establishes it as a REXX procedure. When the "/" string isn't found, the interpreter assumes it to be a DOS-style batch file.

The REXX Comment Statement What is this "/"? The REXX language supports the use of comments, informational text not actually participating in the program's execution. You can place a comment anywhere in a REXX program. But, as we saw in the last paragraph, the OS/2 implementation requires an initial comment, on line 1, column 1. The symbol pair "/" starts a comment, and "/" ends it. For example:

```
# Code attribute.  
/* This comment in REXX */
```

Comments, as well as all REXX keywords, may be coded in upper, lower or mixed case and, in general, you may start coding REXX in any column of a line. The language allows indentation that make programs easy to read and comments may be placed on lines with actual REXX instructions. Also, unlike the batch file's "REM statements, used for comments, REXX comments may be continued on one or more lines. As in:



Eloy O. Cruz-Bizet


```
/* REXX sure is easy but since this is my first
REXX program I'd better document it well. */
```

Converting batch file comments consists of removing the characters "REM" and enclosing the remaining string between "/*" and "*/" pair.

REXX Labels and Branching

In languages that support them, labels are used to designate branch destinations within programs. The colon ":" is used in batch files as a prefix to a label name. Its format looks something like ":CHKVAL" or ":START". To convert label names to REXX, simply shift the colon to the end. Our example ":CHKVAL" becomes "CHKVAL:" and ":START" becomes "START:".

```
/* A REXX Program */
START: /* This is a label in my REXX program */
a = 23 /* assign a initial value */
signal chkval /* jump to chkval */
b = a/32
chkval: /* We come here to check values */
if a > 23 then ...
```

Instructing a program to skip one or several instructions either unconditionally or when certain conditions exist, is called branching. In some languages its called "jumping". Batch files use the "GOTO" directive to do this. GOTO specifies a label name as the destination of program flow. In contrast, REXX requires substituting the REXX "SIGNAL" keyword for "GOTO". Match this change to a corresponding change to the colon on the destination label, as described above, and the jump will work well in REXX.

REXX and the OS/2 System Commands

By design, REXX intrinsically accepts OS/2 system commands, such as XCOPY, DIR, REN, MKDIR etc. REXX lets OS/2 handle them when it determines they are not valid language keywords, expressions, or labels. (see Table 2). OS/2 commands are valid statements within a REXX procedure. Your knowledge of OS/2 system commands is immediately applied to the new programs you create. This is how it's done:

```
/* A REXX Program */
START: /* This is the start of my REXX program */
a = 23
signal chkval
b = a/32
chkval: /* We come here to check values */
if a > 23 then
do
MKDIR L123GOS2
b = a/15
end
```

In this example "MKDIR L123GOS2", an OS/2 command, is inserted amidst some REXX logic. This statement although not a REXX keyword executes normally because the interpretation process knows it's intended for OS/2. OS/2 executes the MKDIR on behalf of REXX and returns to the REXX program at the instruction

immediately following the MKDIR command. But there are times when "naked" commands cause some difficulty.

Problems With Special Characters

Common to full featured programming languages, REXX performs arithmetic and logical comparisons. For example, arithmetic addition using the "+" symbol, arithmetic division using the "/", logical or using the "|", greater than using ">" and lots of others. OS/2 commands use some of these same symbols as part of normal command syntax. Obviously, this causes conflicts. Guess why the following OS/2 commands, if placed in a REXX program, interpret into expressions REXX considers erroneous:

```
DIR /P /* DIR divided by P */
XCOPY A: C: /* A: and C: are labels */
TYPE letter.txt > PRN /* entire command not
recognized */
```

Despite the severity of the problems, the work-around is elegantly straight forward. Simply place the entire OS/2 command or batch keyword in single or double quotes. Doing so converts the errant command to a string expression that REXX doesn't evaluate arithmetically. It obtains the value of the characters and directs it to OS/2. Here are examples of this simple cure.

```
"DIR /P" /* I want DIR to pause */
XCOPY "A: C:" /* copy floppy to hard drive */
TYPE 'letter.doc >' PRN /* output to the printer
*/
```

Insulating selected parts of troublesome commands, as shown, seems logical, but is not enough in many instances. Let's use the last example to illustrate why. Say, for example, that long after you write this program, you decide that a variable to describe a special type of processing would spiff it up. You declare a variable named TYPE" and assign it a value. "A simple change," you say, and don't expect it to cause much trouble.

```
/* Added to process special cases */
TYPE = 'SPECIAL' /* this is a special run */
```

To your horror, upon execution, an OS/2 error message appears when the following expression is executed:

```
TYPE 'letter.doc >' PRN /* this worked before */
```

What happened? REXX evaluated 'TYPE' as a character string variable expression, replaced its value with 'SPECIAL', evaluating the statement to be:

```
SPECIAL 'letter.doc >' PRN
```

The new command was passed to OS/2, which balked, "SYS1041: The name specified is not recognized as an internal or external command..." at seeing the unknown command name. The moral: always be overly cautious when specifying commands. Play it safe and enclose the entire OS/2 command within single or dou-

COMMENTS, LABELS AND SYSTEM COMMANDS

ble quotes, except variable names REXX may need to resolve.

Summary

That's it. For simple batch files containing only OS/2 commands, REM's, and maybe some branches, your work is complete. Now, just save these changes, remembering to denote the filetype as ".CMD". Execute it normally as you would a batch file; typing its name sans the ".CMD". Numbered error messages from REXX are distinguished with the prefix "REX" and detailed explanations are obtainable in response to the "HELP REXnnnn" at the command prompt.

Your simple DOS or OS/2 batch files are now REXX. As you learn advanced REXX facilities, you may add arithmetic, hex conversion, string and sub-string processing and even play music. Explore the fuller range of REXX capability. It will become apparent quickly why the trivial exercises presented here were well worth the effort. Besides opening the door to enhancing crude batch files with more robust REXX facilities in the future, you have taken a small but significant step towards learning a powerful programming language.

In the next installment of "Replacing Batch Files", we'll cover testing, debugging, and the use of PMREXX and REXXTRY facilities.

Bibliography:

M. F. Cowlishaw. *The REXX Language, A practical Approach to*

Programming, Prentice-Hall, Inc., Englewood Cliffs, NJ, (1990)

IBM Operating System/2 Procedures Language 2/REXX, Users Guide S01F-0283, IBM Corp. (1992)

IBM Operating System/2 Procedures Language 2/REXX, Reference S01F-0271, IBM Corp. (1992)

OS/2 1.3 readers may find a simple introduction to REXX in IBM Operating System/2 Users Guide, Volume 1: Base Operating System, S01F-0289, IBM Corp. (1990) On-line information is available in OS/2 Version 2.0 in the Information Folder "REXX Information" (1992).

Table 1. OS/2 Batch File Directives

APPEND	ENDLOCAL	GOTO	PAUSE	SETLOCAL
CALL	EXTPROC	IF	REM	SHIFT
ECHO FOR	PATH SET			

Table 2. REXX Language Keywords

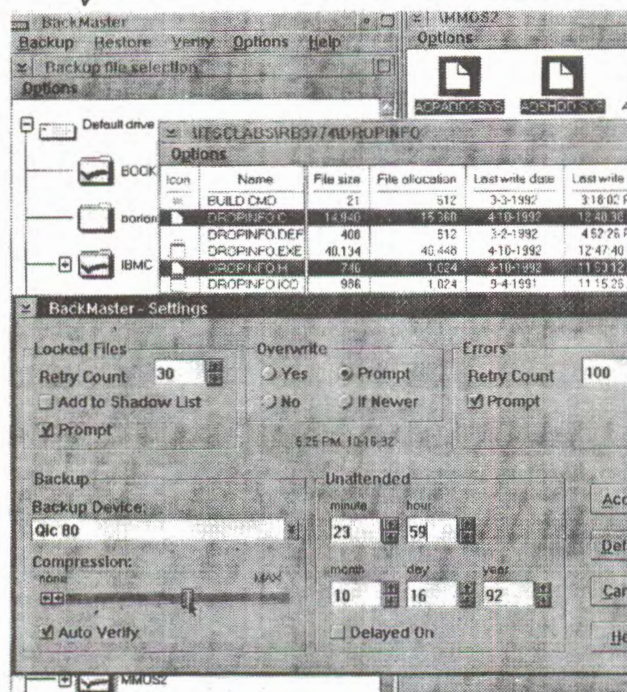
ARG	ELSE	INTERPRET	NUMERIC	PROCEDURE	QUEUE THEN
CALL	END	ITERATE	OPTIONS	PULL	RETURN TRACE
DO	EXIT	LEAVE	OTHERWISE	PUSH	SELECT WHEN



BackMaster

Special \$49.95

OS/2 2.0 32-Bit PM Based Backup Utility.



Supports FAT, HPFS, Long Names, and Extended Attributes.
Variable Data Compression, Printed and On-Line Documentation.
Backup/Restore to Floppies and QIC-40/80 floppy Tape Drives!
Incremental Backups: Wildcards, Time/Date, & Graphical Selection.
We Support CMS JUMBO Tape Drives.
Unattended / Delayed Operation...
Runs Transparently in the Background.
Exploits Multithreading for better performance.
List Price \$79.95, Special \$49.95 + \$5.00 Shipping until Dec 25.
Texas Residents add 7% Sales Tax (\$3.49).
For More Information Call or leave a message on our support BBS.
Call our 24 Hour Support BBS 2400-14,400bps, for Demo Version.
To Order; Call, or Send Check, money order, VISA/MC to:

MSRDevelopment

Rt 7 #6409 Nacogdoches Tx, 75961

Ph (409) 564-1862 --- BBS (409) 560-5970

CALLING REXX

Steve Price

Why would you want to call a program written in the Restructured Extended Executor (REXX) language, when you are writing a program or application in some other language? There are two common reasons. First, each language has its strengths and weaknesses, and writing the application in a mixture of languages may be much easier than writing it all in a single language. Second, REXX is a good macro language — powerful and also easy for non-programmers, novice programmers, and experienced programmers to use for extending and customizing applications. To allow REXX macros to be used from your application, you have to call the REXX programs.

So how do you call a REXX program? REXX on OS/2 is implemented by an interpreter, so you do not call REXX programs as you would compiled programs. Instead, you call the interpreter, and tell it what program you want run. You can pass in the program name and parameters and get back results.

First, a few general points about the interpreter, and then we'll cover the interface to it.

The interpreter is contained in REXX.DLL, with supporting routines in REXXAPI.DLL and REXXINIT.DLL, and some external functions in REXXUTIL.DLL. The interpreter is re-entrant. Multiple threads in the same or different process may run REXX programs simultaneously.

The interpreter is not a Presentation Manager (PM) program. Screen I/O is done in line mode. So if you call it from a PM program, you must intercept the line-mode I/O from the interpreter and any other programs called by the REXX program. A full discussion of how to do that is beyond the scope of this article. For an example of doing this, see the PMREXX sample program in the OS/2 toolkit.

The REXXStart Interface

The entry point to the interpreter is called REXXStart. Like all of the REXX programming interfaces, it is prototyped for C language programming in rexxsaa.h, and linked to by using rexx.lib. These files are included with the OS/2 toolkit. This discussion will give the details about REXX in OS/2 2.0. If you are building a 16-bit program, the interpreter entry point in OS/2 1.X is called REXXSAA, which works much like REXXStart. In 2.0, REXXSAA is also supported as a 16-bit interface, by means of a 16-to-32-bit thunk in the 2.0 interpreter.

REXX deals with unlimited-length strings, so it needs a handy way to represent these strings. REXX allows null characters in its strings, so null-terminated strings are not the answer. Instead, a simple structure called RXSTRING is used. This contains two fields: a pointer called strptr which points to the start of the string, and an unsigned long integer called strlength which gives the length of the string.

What are the parameters to REXXStart and what do they do? Here they are:

ArgCount (LONG) — input

ArgCount is the number of input arguments you are passing to the REXX program.

ArgList (PRXSTRING) — input

Arglist is a pointer to an array of RXSTRINGs, each of which contains one of the arguments to the REXX program. Each argument may be of any length, including

CALLING REXX

zero length. Also, some or all of the arguments may be omitted by making the RXSTRING empty (the strptr field NULL). There is no limit on the number of arguments you can pass to a REXX program. The interpreter will ignore the values of these arguments, with one exception. If CallType (see below) is RXCOMMAND and the first argument is //T, that tells the interpreter to tokenize the program only, without executing it. This is primarily for use with the Instore parameter (see below).

ProgramName (PSZ) — input

ProgramName is the name of the REXX program the interpreter is to run. Normally, the interpreter does the job of locating the program and reading it in. If ProgramName contains drive and directory information, the interpreter will use it. If not, the interpreter will search the current directory and the PATH. Also, if no extension is provided, the default extension .CMD will be used. ProgramName also provides the name string returned by the PARSE SOURCE instruction.

Instore (PRXSTRING) — input

Instore is a pointer to an array of two RXSTRINGs. This allows you to call a REXX program that you have already loaded into memory, avoiding the performance penalty of disk access.

Instore[0] identifies a buffer containing the source of the program. Lines within the source are identified by the location of carriage return/line feed pairs. An end-of-file character at the end is optional.

Instore[1] is used for the tokenized image of the program. What is a tokenized image? When running a program, the interpreter starts by scanning and parsing the whole program. For each clause, one or more tokens are created and saved to describe the clause. When this step is complete, the tokenized image containing all the information needed to execute the program is organized to allow speedy execution.

When Instore is being used to run a program, Instore[1] should be initialized with a zero strlength and a NULL strptr. When the program completes, the interpreter will update Instore[1] to identify a buffer containing the tokenized image. The interpreter does not keep track of that buffer; it is up to the calling program to issue DosFreeMem to release when it is no longer needed.

The particular advantage of using Instore[1] comes when you are going to run the same program repeatedly. On the second and subsequent calls to REXXStart, you pass the tokenized image back to the interpreter. This saves the interpreter from having to perform tokenization. In fact, when Instore[1] is provided, Instore[0] is not used and may be empty. The one exception to this rule is that the SOURCELINEQ built-in function uses Instore[0].

One final feature of Instore lets you exploit the REXX MacroSpace, where REXX loads programs into memory for quicker execution. If Instore is specified, but both Instore[0] and Instore[1] provide a zero for strlength and NULL for strptr, then the interpreter will run the program from the MacroSpace. If the interpreter fails to find the program named in ProgramName in the MacroSpace, it will search no further and return -3, indicating "program is unreadable".

One word of warning: The format of the tokenized image is not a defined interface. Do not make any assumptions about how it is structured or what its size will be. The format differs between releases of OS/2 and can even differ between service levels, so if you save a tokenized image and attempt to use it with a different level of the interpreter, it may fail.

EnvName (PSZ) — input

EnvName gives the name of the initial command environment that will be in effect when the program starts. The default is CMD, for standard OS/2 command handling.

CallType (LONG) — input

CallType indicates whether the program is being called as a command, function or subroutine. This controls whether the PARSE SOURCE instruction will return COMMAND, FUNCTION or SUBROUTINE for the second token. The file rexxsaa.h defines three symbols, RXCOMMAND, RXFUNCTION and RXSUBROUTINE for use as the CallType parameter. Generally, commands have only one argument, but this is not an enforced restriction.

Exits (PRXSYSEXIT) — input

Exits controls which, if any, exit routines are called during execution. These exit routines complement or replace the interpreter's handling of events like initialization, termination, command handling, external function calling, and screen and keyboard I/O. Space does not allow a full discussion of exits here.

ReturnCode (PLONG) — output

ReturnCode is the return value from the REXX program, converted to a binary integer, if that return value is an integer in the range -2**15 to 2**15-1. Otherwise, it is set to zero.

Result (PRXSTRING) — input and output

Result is the return value from the REXX program, in character form. If Result specifies a buffer on entry, that buffer will be used to return the value, providing the buffer is large enough. If the buffer is too small or not provided at all (indicated by making Result empty upon entry), the interpreter will return the value in memory allocated by DosAllocMem. The calling program then has the responsibility of freeing that memory with DosFreeMem. Note that the interpreter does not add a trailing null to this string.

Summary

REXXStart enables your application to call REXX routines from any OS/2 process. You can pass multiple arguments to the REXX program, and set it up to run without any disk access.

REXX has other interfaces in the areas of exits, external functions and subroutines, command handling environments, access to the REXX variable pool, the REXX MacroSpace, and asynchronous request services, such as raising a HALT condition.

REXX is contained in OS/2 Standard Edition 1.3 and Extended Edition 1.2 and 1.3, but not in Standard Edition 1.2. It is also included in OS/2 2.0 and optionally installed. ♦



LES BELL INTERVIEWS MIKE KOGAN

Dr. Michael S. Kogan, independent consultant, was formerly a lead OS/2 developer, lead designer of OS/2 2.0 and coauthor of *The Design of OS/2*. He received his B.S. from Emory University in Atlanta GA, where he worked in Berkeley UNIX and CP/M environments on VAX and 8080-based systems. He received his M.S. and Sc.D. Degrees from Nova University in Florida, where he is also a visiting professor. His dissertation research examined the motivation and design of 32-bit OS/2, while exploring the design and implementation of future system elements.

Dr. Kogan worked for IBM Boca from 1984 to 1992 as an advisory technical staff programmer. In 1987 he began three years as lead designer for 32-bit OS/2, focusing on the definition of 32-bit API and DOS compatibility architecture and implementation strategy. He published many articles on OS/2 and is credited with over a dozen software inventions for technologies developed for the 32-bit OS/2 system. He currently specializes in providing OS/2 consulting and education services, teaches technical seminars and participates on industry-wide panels and consortia.

LB *Things must have been pretty euphoric at the end of April when OS2/2.0 came out.*

MK Yes. In fact, a lot of people took a month off and just last week we had a big celebration for 900 people at a five-star hotel in West Palm Beach. It was incredible. It was Lee's night out and Jack Kuehler showed up. They gave away one week vacations to 400 developers and their spouses. Cruises, Lake Tahoe, San Francisco, Martha's Vineyard, Bourbon Street — and everything taken care of: transportation, hotels and spending money.

LB *I hear Lee (Reiswig) actually pulled a spectacular stunt after the gold code shipped. What was that about?*

MK After we shipped, we had area meetings to celebrate and Tommy had this big box in his hand. They opened it up and it was a window. He smashed the window and behind it OS/2 lit up. Then, there were the video tapes of Jim Cannavino skiing downhill. Coming off a jump, he crashes through a window. Our marketing is starting to come around with some TV spots and I think we'll be seeing some OS/2 commercials in the fall.

LB *From your perspective, is the future pretty well assured for OS/2?*

MK The critical issue is: When will a version of Win32 appear on top of a system that can compete in terms of resource and applica-

tion base with OS/2 2.0? The way I read it, the NT beta I'm going to get next week in San Francisco is supposedly full-function, so people will finally begin to see how much disk space and memory it takes up — they are moving to a 12 minimum and recommending 16 megabytes of RAM. Furthermore, they'll go through a full beta cycle. The first NT with Win32 on it will ship sometime in 1993, no-one knows when. Win32 and NT are not going to be able to compete with OS/2 on the client end of the desktop, with regard to memory, amount of drivers, amount of disk space consumed and just basic performance. As for the number of apps it runs, their security will actually handcuff them and prevent them from running the worst DOS apps and the worst Windows apps.

LB *Because your kernel is written in Assembler?*

MK No, our kernel is written mostly in C. Before I answer this question in depth, I would like to elaborate a little more on your previous question. Where will Win32 apps compete with OS/2 32-bit apps? That really looks like Win32 on some version of DOS and then you are going to have (I don't know what it's called) Windows 3.2 where they integrate the Win32s with 3.1 and potentially with the Windows for Workgroup stuff, as an interim step until they get late next year to DOS 6 with multithreading and put the real Win32 on that. You have to ask yourself where Microsoft is going with this. The original goal of being able to recompile a 16-bit

winapp to Win32 was not met and that's why Win32s was introduced.

When you look at the migration, you see a window of opportunity for OS/2 to build the 32-bit applications' momentum between now and the time that an incarnation of Win32 on a client machine like OS/2 arrives. Then you look at the application migration scenario. To get from Win16 to Win32 apps, you have to change many things; in Windows the message order changes because your message handles and window handles and all those things go to 32 bits and there's not enough room for the data in the message. Sometimes, you get two messages where you used to have one.

The way the state for the multiple input queues are managed changes applications that are sensitive to how the focus is gotten and given away. Once you have done all those things and you have got a flat model Win32 app, you still have not changed your memory allocation to use the real virtual memory API's. That's just a cheap mapping with local and global allocs. Now you have to go and use the real memory management interfaces — you have to rip out your DDE stuff and redo it if you have depended on memory sharing in the 16-bit Windows world. You have to change all your file system calls to real file system calls; you have to do a real awkward shared-memory scheme for Win32 that deals with mapped files, you have to add multithreading and you have to add interprocess communication. All those things have to be added to your app.

Win32 is an OS/2 2.0 clone. If you compare the API function for function, OS/2 2.0 is much easier to use than Win16 because it's better-behaved. With separate address spaces, protection, it's everything OS/2 is plus a little bit more. Those things are coming in OS/2. When you look at the migration stories, I think those are some significant points to bring up.

LB *Are you saying that implementing an NT compatibility into OS/2 is under consideration?*

MK Definitely. I think when we do our Win3.1 support we're looking at providing some kind of Win 3.1 enhanced mode support so that we have the option of running Win32s applications. It's not a question of, "can you technically?" It's a question of "does management want you to do it?"

Under the covers, everything in the kernel, except for some of the file systems codes and the device drivers is 32-bit and most is written in C. All the task and memory management, interprocess communications and everything like that are already in 32-bit. Outside the kernel, now that we are shipping the 32-bit graphics engine, the only parts that are 16-bit in the guts are PMWIN and the queue calls. Our next major point release will be 3.1, the 32-bit engine, more OEM support and an update with bug fixes. The next major point release next year will be a 32-bit PMWIN with multiple input queues.

We are not introducing a 32-bit device driver model yet. Our goal right now is to use Mach as a technology for a microkernel, enabling us to get levered onto RISC systems. Up my sleeve is a trump card called symmetric multiprocessing for X86 using our current code base. I would love to demonstrate that on the same system NT is running on side-by-side at the next Comdex.

But I view the Mach stuff as a lever into the RISC world, when it's needed. When you can buy the RISC workstation, which is

Is your LAN Serving you? Or are you Serving it?

THE SOLUTION IS
Getting the right training from the start:

- ▶ LAN Server 2.0 for Administrators
- ▶ Advanced LAN Server Performance Tuning

AR&D also offers OS/2 programming classes in C, REXX, ObjectVision and provides LAN Server installation support.

Are you still
using SneakerNet?!!!



Classes on our site, or yours.
Call for registration information.

Adaptive Research & Design Co.

12000 Biscayne Blvd., Suite 203
Miami, Florida 33181
Phone: (305) 899-0070
Fax: (305) 892-8669

more powerful than the Intel platform, for less money, it becomes interesting to have applications to run on and a system that supports it. Portability is really the driving force now. The application programmers want portability at the API level so they can recompile for RISC. They've got that today with OS/2.

The underlying implementation has to be there when the hardware volumes and the market looks like it's ready to move in that direction. Granted, we are missing a piece at the very high end. SMP's real application is not on a file server but on an application server. Lotus Notes server is a good example, because Lotus Notes does database-like things and runs out of mippage when it's supporting a lot of users. It's not I/O constrained.

Concerning portability, Mach is going to become the underpinning of OS/2. We are trying to drive Mach into the Taligent system, so it will run on top of a Mach microkernel rather than the one it's running on now. There's a ton of research going on out there on Mach — fault tolerant, real time, secure Mach — funded by the U.S. Department of Defense. While NT is very Mach-like, it's proprietary and that's a big difference. Mach is open, and that seems to be the base for systems in the future — when you can have multiple-operating-system personalities co-existing. But to run Mach and NT you are really talking about a client-server system, and entry-level systems don't run those. In fact, NT won't even run on the majority of 386's out there, because it won't support a B1 stepping-level 386. The entry NT system is starting to look like a 33 MHz 486 with 12 or 16 megs of memory, which is a tough requirement.

LB Will Mach microkernel-based implementation be a 2.X release or in OS/2 version 3?

MK We don't know yet. We don't expect it to be ready until '94, though we may have some demonstrations earlier.

LB Have you been out talking to the developers recently — the eternal developers?

MK Yes. I did a seminar in San Francisco a month ago and one in Toronto a couple of weeks ago. All the developers are charged up and some of the major vendors are moving to OS/2 in a big way — all the Lotus apps, all WordPerfect apps — just about everyone but Microsoft. In the Windows market, the squeeze is on. Microsoft apps were out there first and they're bundled with everyone's systems, so it's difficult for a lot of those leading vendors to penetrate the Windows app market. But the OS/2 market is obviously going to be a large one. It's going to sell a million and a half to two million copies this year. The market is there and there's money to be made in it.

LB When you talk about their apps, without giving away any proprietary information, are they initially looking at simply porting their OS/2 1.X apps across or are they doing a major redesign for Workplace Shell?

MK Workplace Shell does not require a major redesign of a Presentation Manager application. Instead of just using suffixes to drive the linking up of files with the application that runs them, you build a small class library to interact with the workplace and add your file type as a class to the system. When the user manipulates those icons on the user interface, your class library gets control and

you're able to do some fancier things, like context menus. For example, a spreadsheet file will have the icon of the spreadsheet program. Some of those things can be done today without any application support inherent to the shell, but by writing a class library it can be done a lot better. Right now the workplace shell and the System Object Model are an open-ended application, called the Shell, which you can tie into. It is continually being enhanced to deal with distributed environments and things like that. When that happens, the object model will be enhanced inside the PM as well.

LB Is that going to be on AIX?

MK It depends on whether the next release of AIX beats the next release of OS/2. SOM is a cross-platform and the really neat thing about SOM — the thing that excited the languages people — is that this allows someone who is writing in a non-object oriented language, someone who is not in C++, or in Smalltalk, to get access to the object-based world. But it doesn't do much for someone using an object-oriented programming language. As for Pink competing with all this — Pink is a different ball of wax altogether, where you throw away the whole procedural programming model and use different languages, object-oriented programming languages, in an operating system built with object technology. It's not a competitor, it's the next step after today's procedural-oriented systems.

Zippy Object Oriented Memory™ The ONLY OODBMS for Smalltalk/V priced under \$1000 that delivers Persistent Object Storage on Disk via a Zippy B+Tree Database Retrieval Engine!

available for Smalltalk/V (Win, PM, Mac & 286)
All Platforms \$199.95
Source Code Included!



HIERARCHICAL APPLICATIONS LIMITED

"Working to Make Smalltalk a Better Place to Live"

(512) 837-2117 or (800) SKY-PAGE Pin 5466843#

12407 North Mopac, Suite #100-266, Austin, TX 78758

INTRODUCING: dBase Read-Only File Streams for Smalltalk/V (Win & PM) \$49.95 for Smalltalk-80 Rel 4.0 \$149.95 Source Code Included!

LES BELL INTERVIEWS MIKE KOGAN

LB *Is it like Smalltalk?*

MK I don't know what the personality of Pink is, but you wouldn't write a program to call the Pink "API's" by using C. What you would do is use an object-oriented program and access the methods exported by the system for objects in the system.

LB *Getting back to the kernel — at present the file system drivers are 16-bit. What are the plans for porting them to 32-bit technology for a possible performance hike?*

MK Actually, there's not a performance hike at 32-bit and that's why we elected to leave the 16-bit in the near term for this release. But, while our file system is 16-bit code, it does have 32-bit read and write interfaces. It's not limited to 64 K transfers at a time. Secondly, when you go to do file I/O, the first thing you do once you've gone into the kernel and gotten to the file system is determine whether you have a cache hit or not. If you have a cache hit, you return and it's very quick. If you have a cache miss, you do I/O and it's super, super slow. Going 32-bit would merely speed up the time it takes to determine whether you have a cache hit, and that's not a significant delta in performance, looking at the throughput of file management and I/O in general. That's why we chose to leave that in the near term.

In the long term, however, we are looking at the file system technology and going for a 32-bit installable file system model comparable to the UNIX environment with the virtual file system technology. It's similar to IFS, but more refined and a little bit more mature. We are prototyping that on the Mach platform, with 32-bit device drivers as well, because we want to guarantee that we can take these driver models cross-platform with us. The device driver talks to the operating system and to the device. We ought to be able to come up with some interfaces, like UNIX, so that device drivers can be taken from platform to platform, where the same operating environments are running, or the same operating environments that are using the same microkernel. All of those factors come into play and that's the one area where Mach is weak. It doesn't have much for I/O or device management.

LB *The next major revamp of PM — that's the one with the multiple message queues and de-serialised models.*

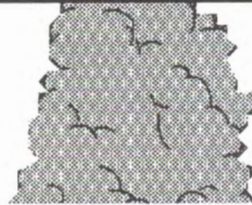
MK And PM OLE, though I don't know whether it's going to be called OLE. The thing about object-linking and embedding is that what's in Windows today — the 1.0 stuff — wasn't good enough, because it didn't have system-object management, like SOM.

LB *How do you map or re-map between OLE on the one side in the Windows World and SOM in the PM world?*

MK That's not a major problem. The only problems with the semantics of OLE come in the multitasking area, like with DDE. When you have a client receiving a message, it's got to be in the foreground for the conversation to run accurately, and we had to put some hacks in the component called the DDE server to make that run properly. OLE is built on top of DDE and will also be capable of integrating or linking a PM app with a Windows app, hopefully in a more transparent manner. It will also interact with the System Object Model for managing objects on the PM side of the world.

The major PM enhancements that are coming are 32-bit PMWIN, multiple input queues and object linking.

CPU Monitor Boosts OS/2 To The MAX



CPU Monitor™ gives a real-time boost to your OS/2 1.2, 1.3, and 2.0 environment. With CPU Monitor, you can display real-time performance

and CPU utilization data for all processes and threads; detect and stop invisible, runaway, and background programs; suspend and resume threads, and dynamically change thread priorities for any of your Presentation Manager applications. You can tailor the displays exactly to your liking since CPU Monitor's displays are fully configurable.

For a limited time, CPU Monitor is only

\$99.95

Start getting maximum performance out of OS/2. Order CPU Monitor today!

BonAmi Software Corp.
60 Thoreau St., Suite 219
Concord, MA 01742 USA
(508) 371-1997

Now for OS/2 2.0!

BA's CPU Monitor		
File View Commands Options Help		
Name	CPU Utilization	Time
IdleTime	53%	
MONITOR	12%	
CMD	11%	
IdleTime	9%	
COUNTER	9%	
LOWMEM	3%	
BACK	3%	
PMSPOOL	0%	
PMEEXEC	0%	
HARDERR	0%	
PMSHELL	0%	
(kernel)	0%	

LB What's the significance of the multiple-input queues for developers? Is it going to break any interesting applications due to the change in the way that messages are processed?

MK No. Under OS/2, and under Windows, multiple-input queues really don't make a difference, except to applications that are sensitive to the states involved when you get or set the focus. Applications that go out and grab the focus rather than let the system give it to them can have problems, because certain API's like WinSetFocus() and WinGetFocus() will return different states. That's where the compatibility problem came in for Windows.

LB It happens when applications are doing what — like popping up message boxes for error handling, that kind of thing...?

MK I don't have any apps that have problems with the PM. The messages are slightly different than the ones Windows has for focus management. Scott Klieger, one of our gods of PM, was working with Microsoft on the multiple-queue model way back when we were still communicating on these things. He understands all the issues and the code that needs to be written. It's a sizeable amount of work. With the rush to ship, we rarely have a chance to catch our breath, but plans for developing new products have begun. We've got one group working on multiple input queues and another, on the 32-bit PMWIN, which we're planning to use on our Mach system and the X86 base source. *We want some code re-use.*

LB Are the multiple input queues going to eliminate a lot of the cases where you get the bourglass appearing?

MK Yes, and it will also free up the user interface when you have an application not responding to messages; the system is better able to manage that. It's an integrity statement more than a performance statement.

LB And for the app developers, it's business as usual.

MK If you were multithreaded before, you'll probably stay multithreaded. Multithreading has always been more than just a way to get around that problem. You don't have to multithread your app, though it's still a good idea, because the apps that did that have lots of other processing.

LB Last year when you were out here, we spoke briefly about some other things which were being discussed in the labs — for example, floating point co-ordinates space in PM. Is that happening?

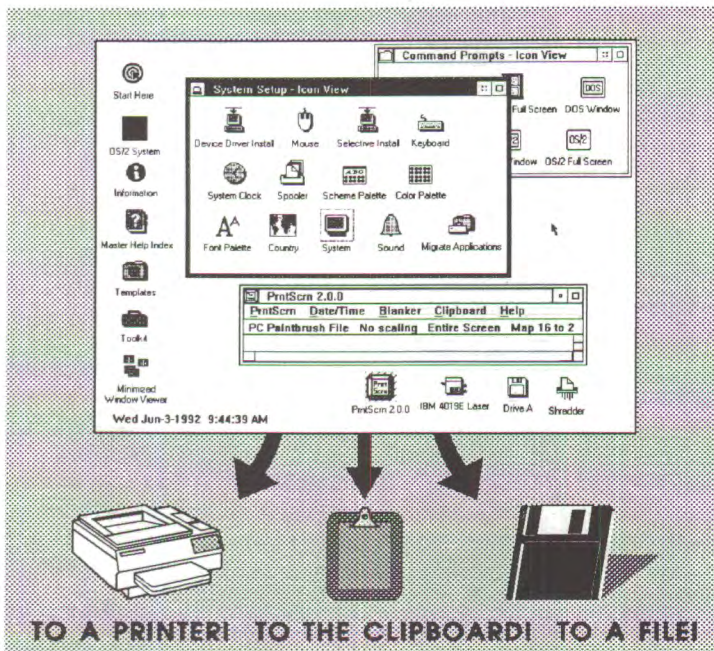
MK It's not happening right now.

LB Is it happening at all?

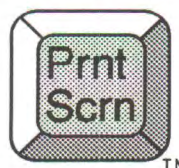
MK It's down the list. At present, 32-bit PM and multiple input queues and object linking and embedding are the three high priority items for the Presentation Manager.

Another PM project that is being developed, though I'm not sure when it will ship or what kind of incarnation it will be, is Distributed PM, a la X-Windows. Our advantage with PM is that we already

CAPTURE YOUR SCREEN



MITNOR Software
28411 E. 55th Street
Broken Arrow, OK 74014
Phone: (918) 357-1628
Fax: (918) 357-2869



PrntScrn™ is THE One-Step Screen Image Utility for copying a graphics image from the Workplace Shell screen to a LAN or locally attached printer, to the Clipboard, or to a disk file. The screen images appearing in this publication were captured and processed using the **PrntScrn** utility. The capture operation can be initiated from the keyboard (by pressing the print-screen key), from the mouse, or from another program. The captured image area can be the entire screen, current active window, current active client area, selected window, selected client area, or specified rectangular area. Disk file formats include PM Metafile, PM Bit Map, PCX, TIFF, GIF, and WPG. The "Color Mapping" feature provides the control needed to produce quality printed images from color screen images. **PrntScrn** has clipboard Viewing, Printing, Exporting, & Importing capabilities for Bit Maps, Metafiles, and Text files. **PrntScrn** includes a Screen Blanker and a digital Date & Time "information bar". **PrntScrn** is priced at \$115 and is available from major software resellers, or contact MITNOR Software.

Windows, OS/2 & Unix

Consultants Programmers Systems Analysts



800-862-2211
Ext. 50 #

Technologies, Inc.

We provide on/off site consultants and programmers for short or long term assignments, who are specialists in, but not limited to, application design and development in the following areas:

Microsoft Windows
Presentation Manager
Communication Manager
Database Manager
LAN Manager
X-Windows
UNIX Communications



Some of the services we can provide you with:

Product Planning
Cost Analysis and Projections
Project Leadership and Management
Requirement Analysis
Functional Specifications
SAA/CUA Application Design
System Testing and Verification
System Design and Development
Host Front-end Design



Come to us first when you need software engineers to design, code and implement your Microsoft Windows, OS/2 and Unix applications.

Ceres Technologies, Inc.

8903 Glades Road Suite L-9234 Boca Raton, FL 33434-4021

LES BELL INTERVIEWS MIKE KOGAN

have an application today called PM/X. That runs on top of the TCP/IP package and allows you to run an X-Windows server on your OS/2 desktop. It takes those X messages coming in that are bound to the X server and that's a display driver basically an X server, then converts them into the appropriate PM calls and you see your X app running on the OS/2 desktop.

We're going to have the opposite possible and we'll distribute Presentation Manager applications where, for example, you have what's called the X client — the part of your app that really does the work — running on the applications server, and you have your PM server running on your workstations. The server then takes the client and interacts with him, displaying the client's desktop on your system. What you have today under X Windows with the UNIX systems is load balancing. For example, when you go and start Lotus 1-2-3, the system could go on the LAN and say, "How many different Lotus 1-2-3 clients are installed on this LAN and which one has the greatest amount of processing power for me to transfer." Et cetera. It's called load balancing.

It also provides you a foothold inside the OSF world, with UNIX interoperability, the Distributed Computing Environment portions of OSF, the RPC specifications and all the other features that come with that. It will be incorporated into OS/2. SAA is embracing that so you'll have interoperability between all systems under the IBM umbrella throughout the world.

LB Would that allow me to implement the distributed PM server on either an OS/2 1.X machine, or even the old DOS machines or will I need to be an OS/2 2.0 box?

MK You could use the 1.X — it's similar to DesqView/X. With DesqView/X, you can use an X-Windows program that's running on anything else in the LAN on your DOS system. Or DesqView/X can take a DOS program running on that system as a client and turn it into a standard X client, so the DOS program can be run by a RISC workstation somewhere else in the LAN. It goes both ways with X.

LB Another thing that occurred to me was the upcoming release of the P5. That's got some interesting possibilities that would apply in the multi-tasking, multi-threaded area.

MK Does it? I don't think so. It only has a couple of interesting capabilities. The compiler writers will be able to take advantage of the multiple pipelines inside the processor and generate code that runs super fast. Intel's released those specifications to all the major compiler folks, which is crucial. You want to have a version of your system compiled for the P5.

In a general way, you might say the P5 does come with enhancements that will allow higher performance, virtual 86 machines to be created with a greater integrity than today. They have a hardware virtual interrupt flag, for example. You don't have to use IOPL to virtualise it any more and there are some other things that allow direct dispatch through the interrupt table of a virtual 86 mode app. It's the same when you do an INT21 on the 386/486 — it goes to ring 0 no matter what. Or on the P5 you can set up these bits on a per-interrupt basis that states, "Go through the IVT" or "Don't go through the IVT."

There are a lot of things that will speed up the system by reducing the number of transitions between ring 0 and V86 mode, which is ring 3.

LB *Is there any 486 specific code in OS/2?*

MK Not really. The only interesting thing in the 486 is perhaps selective TLB (Translation Lookaside Buffer) flushing and we really don't take advantage of that, because the nature of the beast is that you end up flushing a lot out of there at one time. We didn't go to the trouble to take advantage of that.

LB *It sounds as though you'll be able to really take advantage of those features in the P5.*

MK There are other important features in the P5. It has a larger block-move instruction than the 486 has, which speeds up internal copy operations and things like that, so there will be some areas where you'll want to go and tweak the code to be slightly different on a 586, plus areas where you'll want to compile everything differently.

LB *So far, we've talked mostly about the client side of the architecture. Are you planning any enhancements on the server side of things?*

MK LAN Server 2.0 Advanced is coming later this year.

LB *That requires a re-write of HPFS386.*

MK It won't need a complete re-write. HPFS386 is not in the current 2.0 source base. The same modifications that we made to the 16-bit HPFS in 2.0 have to be made to HPFS386, as well as some changes to the network and the disk drivers. You have to keep in mind that the server used to run as a ring-3 application, and as a result, there are extra context switches slowing down the performance of your server. We want to drop the server down to ring 0, effectively running in a device drive.

So you have some nasty hooks between the file system, the disk driver and the network driver, and those hooks evolved into this thing called the Extended Strategy Interface in the block driver. For example, where you can have command chains and scatter-gather etc., and we formalised that and are releasing it in a DDK for OS/2. The old HPFS386 code and the old network drivers have to be brought up to date with the rest of the world. That's what going on with the LAN Server 2.1 or 3.0, whatever it will be called. HPFS being 386 or 32-bit assembler is not what makes that beast fast; it's server running at ring 0. There are no context switches for getting a job done when a request comes in from a network.

LB *I'm told the next major enhancement for the network server is the SMP implementation.*

MK Right. We want to integrate whatever we do for SMP right into the server. The SMP does not improve file serving capabilities, SMP is a technology for getting more out of CPU intensive applications, getting more MIPs.

LB *SQL servers? Those kind of things?*

MK Yes. SQL servers and application servers are where SMP will be noticed, because that's where the biggest requirement is right now, as well as engineering and scientific applications and number crunching apps.

LB *How's OS/2 going in that market, which has traditionally been a UNIX mainstay? Is OS/2 making headway there?*

MK Sure. There are people in Stanford using OS/2 as a control program for some laser projects — electron splitting I believe they're using OS/2 to run the entire control program. There's a lot

of that going on, and with 32 bit there are a lot of people who are porting applications.

I used to work in the engineering and scientific fields when I was in college. We once took these huge FORTRAN programs, downloaded off tapes from Sandia and JPL, and recompiled them on the FORTRAN compiler on an IBM System 9000 — a Motorola 68000-based box. Those apps always resisted coming to segmented architectures for the normal reasons. A 32-bit flat model allows people to re-target OS/2 and turn the desktop into a place where they can get some work done now. There's a lot of focus from laboratories on OS/2 but they are also looking for SMP — a lot of them right now are running UNIX with Mach underneath. OS/2 is making inroads, except where major MIPpage is required.

LB *I was once doing some work for a client designing instrumentation and data acquisition and control gear using OS/2.2.0 and he asked me about the real time capabilities, which led me to ponder the actual difference between a real time operating system and a system like OS/2, which is pre-emptive and multi-threaded but isn't real time. What's the real distinction, and is OS/2 suitable for real-time work?*

MK Yes. The real differences are the requirements in the real-time system and that you have things like deadline scheduling. Things must happen. We can guarantee a time critical thread will be dispatched within so much time of its becoming ready, but that's not

Learn OS/2 by Simply Watching TV!



As a leader in video training, we are proud to announce the addition of IBM OS/2 2.0 to our complete line of learner friendly videos.

Getting Started with OS/2, 2.0 \$39.95

This video tutorial will take the first-time OS/2 user through the many setup and installation options. Also learn the basics of the OS/2 environment.

Productivity with OS/2, 2.0 \$49.95

This professional training tool will increase productivity. Get the most from OS/2 using this clear, easy-to-follow video presentation.

(add \$5 for shipping)

**Get Both
for \$75**

**Call, send, or fax
your order to:**

5 S. Vann Street, Pryor, OK 74361
918/825-6700, 918/825-6744 (fax)
800/842-4723 (800/VIAGRAF)

ViaGrafix

Computer Training Videos

"The Best in the Business!"

enough for a real-time system like real-time Mach. Mach is a very well refined architecture, very clean. There's a lot of integrity there and the layering is really nice. However, to do real time in Mach, they had to expose interfaces for very high granularity timers, and expose interfaces for locking memory, because you can't afford to swap in many real-time situations. Those kinds of things have to be done in a way that can live within your architecture. A real-time system and security are very difficult to get together, because the more security you put into a system, the slower it gets. A real-time system is a stand-alone system, where your real-time app has to have some serious privilege to get the job done. OS/2, while suited for many real-time tasks, does not provide the guarantees that a true real-time system does.

LB You mentioned that NT is facing a problem because the security will stop them from running a number of dirty DOS and Windows applications. Won't you have the same problem in OS/2 to some extent in supporting those applications? Can you set a balance there?

MK If you want to run those apps, you don't get the great security. I'm not sure what Microsoft plans to do with NT, but if NT is going to run as much as we run today, it can't be any more secure than LAN Server or Citrix Multiuser OS/2 installed. Both of those programs have access control security like UNIX, and you can't do much better than that. Furthermore, what about DOS applications that go out there and talk directly to the hardware? What about Windows applications that use the magic alias to the LDT that lets them write to it?

NT can't be much more secure than OS/2. Once OS/2 has security, you're going to have options that say, "We can't run this application because of security." So you call your administrator who says "You're the super user. Run it". Everything has a cost and everything has a reward. You have to decide which ones you want and which ones you don't.

LB That's a lesson that users are learning in the marketplace. The Microsoft apps are really cute apps, and very popular. But once you sign up with Windows, you put yourself under Microsoft's control. A lot of people are beginning to resent that.

MK I admit I like Microsoft Word for Windows. It costs me 15 megabytes on my disk. People are saying OS/2 is big! The applications are far larger than the operating systems, even for Windows.

LB With the joint agreements over, is there a new agreement pending between the two companies?

MK I don't know yet. Supposedly, they've resolved all these disputes over royalty rates and the like. From what I've read in the press, IBM gets the money for giving Microsoft access to a large number of patents. The royalty rates Microsoft will get when OS/2 is sold have been settled. I believe we have access to the Windows source code through 1993. But it really doesn't make a difference as far as compatibility goes. Our 3.0 compatibility is based on a Win 3.1 beta kernel, which we've already got under control, and Win32 compatibility. We really don't need the source code, because it is already very much like OS/2. ♦

RimSTAR PM Editor

Special
\$99
Offer

Do you need an Editor that has ... ?

Presentation Manager support: Full SAA/CUA PM Multi Document Interface. Complete column, block and line cutting and pasting from the keyboard or mouse. OS/2 2.0 & 1.3 compatible.

Power, Safety and Speed : Unlimited file sizes, Unlimited UNDO/REDO, and Unlimited number of open files and windows. Complete threaded compiles and error stepping integrated in the editor.

BRIEF Compatible and Configurable: If you use BRIEF you'll feel right at home; a complete BRIEF keyboard compatible layout is included. Powerful keystroke macros and configuration options are available for your own customizations of the editor and keyboard.

DLL version available: Do you need an editor to embed into your applications? This is perfect for vertical applications and OEMs that do not have the time to reinvent the wheel. Call for details.

Valued Priced: The \$99 special is valid till October 15, 1992. Then normal \$129 list applies.

Then you need to Call ...

RimSTAR Technology 32 Hawkes St. Marblehead, MA 01945. 617-631-7398

AN OS/2-PM EDITOR USING MLE

MULTI-LINE EDIT PREDEFINED CONTROL WINDOW

**BRIAN R.
ANDERSON**

MLE (Multi-Line Edit predefined control window) is a powerful control window which is available to OS/2-PM programmers (pushbuttons, radio buttons, and list boxes are also control windows); the initials stand for Multi Line Edit. In a very real sense, an MLE is an object that "knows" how to edit text. This article describes how I used an MLE as the basis for building a simple, yet functional programmer's editor.

Background

You would be well justified in asking does the world needs another editor. This project "happened" due to a unique set of circumstances. I was teaching a course on OS/2 programming, where the students were (rightfully) complaining about the OS/2 System Editor, because it did not display line numbers (they needed line numbers to interpret compiler diagnostics which list syntax errors by line). Just as the course got under way, a labor dispute kept me away from my teaching duties for nearly a week. I decided to use the time productively and thereby give the students a more useful editing tool and an extended example of OS/2-PM programming.

While you may never wish to write an editor, per se, you might want to include some minimal editing capability in a larger project — the MLE provides a perfect mechanism for doing so, with very little effort. Although this article describes using the MLE in the client window, I have also used the MLE in a dialog box to give limited text-editing functions for purposes specific to the application.

Object-Oriented Programming?

I previously referred to the MLE as an object. This is not really an article on OOPS, but many of the characteristics that make OOPS attractive also apply to the MLE. First of all, the Multi Line Edit control window is very high level (as are many of the objects in a good class library). Secondly, the MLE uses a message-based paradigm: you send the MLE messages when you want it to do something, and it sends you messages when it wants to advise you of something. Finally, the MLE can be extended easily (through subclassing) to enhance functionality.

What I mean by the Multi Line Edit control window being very high level is that this object

itself already knows how to interact with the user for text entry and insertion, cursor movement, scrolling, search and replace, cut/copy and paste, and probably a few more that I've forgotten. For some of these features, no extra programming is required at all. For the other features, all that is necessary is to send the MLE an appropriate message indicating what you want it to do. Some MLE features require multiple messages and/or a small amount of additional programming.

Article Scope

There have been many good articles published about the basic structure of an OS/2-PM program (creating standard windows, the message loop, the window procedure, resources, etc.). This article assumes that you are already somewhat familiar with those areas of GUI programming and therefore glosses over them. The two main areas of emphasis for the article are: programming with the MLE (including subclassing), and programming for the OS/2 help system (Information Presentation Facility).

Design Criteria

I wanted the editor to be easy to use, but not overly constraining. I also wanted the code to be simple and understandable, so that I could use it as an example in the classroom. Finally, I wanted good compliance to the IBM Common User Access guidelines.

I decided to limit the size of edit files to 64K — partly to keep the code simple, and partly to ensure that my students coded their projects in a modular fashion (no single C source file should be larger than this). I also decided not to support custom colors or fonts — after all, this is a programmer's editor, not a word processor or a paint program. Since with OS/2, it is possible to start multiple copies of the program and then cut/copy/paste among them, I saw no need for implementing the MDI (Multiple Document Interface).

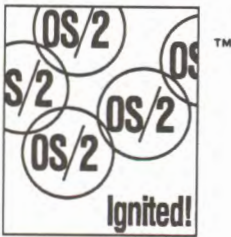
Although I had access to both CASE:PM and Gpf (Presentation Manager code generation tools), I elected to develop the project from scratch — mostly for the practice using the raw API and the SDK.

After several weeks of use, I have found that all of my students prefer the new editor to anything else (previous classes "got by" using DOS editors

EDITOR'S NOTE:

The full, unedited version of this article can be found on the OS/2 Shareware BBS along with the source code and executable.

703-385-4325



Relish[®]

The Premier Workplace Shell Calendar

Ignited!

"Relish is one of the best ways to demonstrate the advanced capabilities of OS/2 2.0..."

Dynamic saving and refreshing across all views.

Drag-and-drop scheduling and categorizing.

Automatic reminders and LAN reconciliation.

Multi-threaded, multi-tasking, client-server design.

That's Reliable.

Ultra-Flexible.

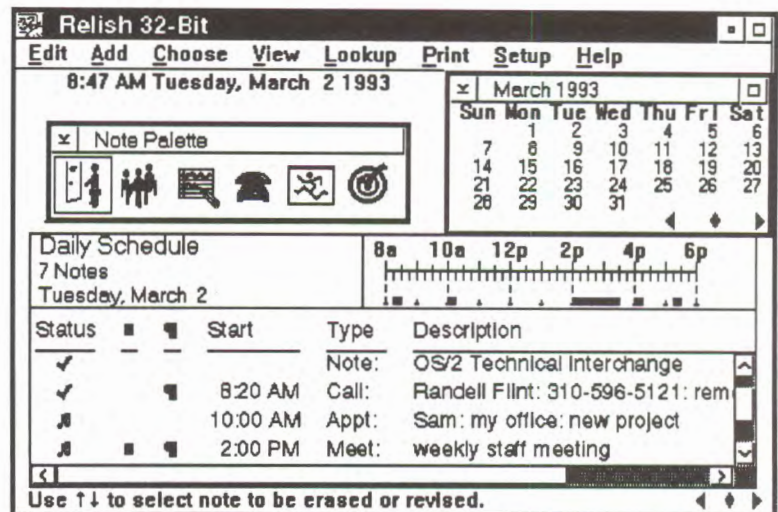
So Versatile.

Cutting-Edge.

Extensive drag-and-drop support

With Relish, drag-and-drop supplements the more traditional methods of using menus and buttons. You have many ways to add to and revise your calendar. Not only can you drag-and-drop within Relish, you can drag elsewhere as well.

Reschedule an appointment by dragging it to a new date or time...
Customize colors and fonts by dragging from the Workplace Shell palettes...
Print the schedule for any day by dragging the date to a desktop printer... It's really that easy.



Relish spreads easily

Time saving features and thorough context-sensitive help are always at your fingertips. Schedule when you want, for any duration, without the barriers of pre-defined timeslots. Search for free time. Reference the integrated telephone/address book. Print anything in view, and calendar-style, too.

Relish 32-Bit for personal calendaring brings an intuitive edge to desktop time and information management. Comprehensive functionality supports the calendar, reminder, and scheduling features. \$189.

Relish Net 32-Bit for LAN-based groups integrates group and resource scheduling with all the personal features of Relish. View your own, another, or several calendars together – updated immediately even as others make changes. Schedule both individual and group commitments. Each 5 user increment \$685.

Your time is valuable – Relish it!

32-bit for OS/2 2.0. 16-bit versions also available. Mixed OS/2 1.x and 2.0 network environments supported.

This is Sundial Systems' 5th year designing and developing quality time and information management solutions for OS/2.



SUNDIAL SYSTEMS

909 Electric Avenue, Suite 204, Seal Beach, CA 90740 USA

Call now for a free taste test.

(310) 596 • 5121

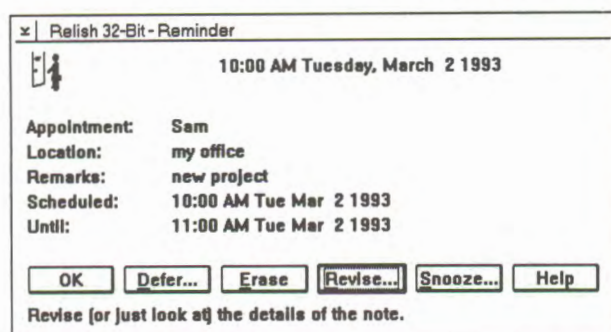
Relish provides a fully integrated approach to desktop time and information management. Schedule everything you need to be reminded about or want to keep track of: Appointments, Meetings, Notations, Phone Calls, Programs to Run, To Do items...



Manage time and tasks with ease

Make notes with pertinent who, where, what, when, and why information. Relish does all the rest, *automatically*. Naturally, Relish can't attend meetings, but it will dial phone calls and run other programs and command files for you. With one note, for example, you can run your file back-up program automatically every Friday at 6 PM.

Reminders are full-featured



Take immediate action from any reminder: ok, reschedule, defer, attach a memo, change the particulars, even change the type (say from a Phone Call to a Meeting). No need to go back to your schedule; no extra steps to waste your time.

Reminders are automatic

Best of all, Relish will always remind you as obligations require your attention. Nothing extra to do when you schedule the commitments. Reminders are assured, no matter what programs you have running. No need even to be running Relish at the time.

View your information

A host of different views provide schedule and other information in a variety of ways. Create ad hoc views by looking up specific information. And see several views at once just by opening multiple windows.

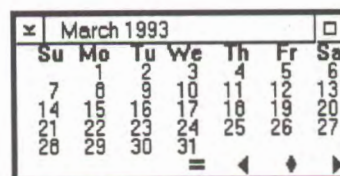
<u>N</u> ow...	Ctrl+N
<u>D</u> ay...	Ctrl+D
<u>W</u> eek...	Ctrl+W
<u>M</u> onth...	Ctrl+M
<u>P</u> eriod...	Ctrl+P
<u>T</u> o do list...	Ctrl+T
<u>O</u> verdue list...	Ctrl+O
<u>P</u> hone book...	Ctrl+K
<u>I</u> nbox...	Ctrl+B
<u>Y</u> ou...	Ctrl+Y
<u>P</u> eople...	
<u>P</u> laces...	
<u>T</u> hings...	

State-of-the-Art

Relish exploits OS/2 to give unmatched flexibility in coordinating and managing your commitments. Ease of use is pivotal. As is reliability. Blends calendaring and information management sensibly into the Workplace Shell environment.

Convenience is the key

Even when Relish is minimized or hidden, a small monthly calendar can remain on your desktop for reference. Any day's schedule is just one mouse click away, with the ability to add to or revise your schedule. What a time-saver!



Technically superior

Relish takes full advantage of OS/2's multi-threading, multi-tasking environment by using a client-server architecture even in single user versions. It works in the background so you don't have to waste time or remember to do things like reconcile and save. A carefully crafted, tried and true design specifically for OS/2. Much more than gadgets and gizmos.

CUA and SAA

CUA compliant. Registered in IBM's SAA™ Catalog.

Unsolicited Comments "Let me congratulate you on an outstanding improvement to what was already a very fine product. The addition of drop-and-drag and increased customizability make Relish even easier to use and more versatile than before." "This is a very impressive application! It looked good at the shows and looks even better close up." "Useful from day one."

AN OS/2-PM EDITOR USING MLE

and/or the OS/2 System Editor). Many students have commented that the OS/2 environment is much superior to anything that they have experienced before (prerequisite programming courses have used everything from a 370-style mainframe to Borland's Turbo C++). What they like best about OS/2 is being able to start compiling in one session, and then immediately go back to editing in another session.

Production Files

Presentation Manager programs require many types of source files. Of course, there are the C source code files and header files. In keeping with my dictum of modular decomposition, I have divided the project into three separately compiled modules: EDIT.C (contains main, the client window procedure, and the subclassing procedure); EDLG.C (contains the dialog box window procedures and supporting code); and EFILE.C (contains the file input/output functions). Each C source code file has a matching header file. Concerning my self-imposed 64K limit: even if all six of these source code files were concatenated, they would not total 64K!

In addition to the C source code, there is the resource file: EDIT.RC (contains the menu, keyboard accelerators, help tables, and dialog templates); the module definition file: EDIT.DEF (contains instructions to the linker); the help text: EDIT.IPF (contains the tag-language help source code); the icon file: EDIT.ICO (contains a bitmap); and the makefile (contains instructions for compiling and linking the entire project).

The total lines of code for the project is about 2500 (including all files mentioned above, except for the icon file). The EDIT.EXE file (executable) is about 25K; the help file (after compiling EDIT.IPF to EDIT.HLP) is about 15K. Installation is accomplished by copying the executable file to C:\OS2, and the help file to C:\OS2\HELP.

Credits

Most of the work for this project is original — however, I am not one to “reinvent the wheel”. The open file dialog box (and associated support code) was lifted verbatim from Charles Petzold's very good book: Programming the OS/2 Presentation Manager. I derived the save as dialog box from the open file dialog box.

How Edit Works

I will begin by describing the overall structure of the program, including a brief description of each function. Since most of the action occurs in the window procedure and through its message interaction with the MLE, I will also describe the purpose of some of the message exchanges. In many instances, standard messages sent by PM to the client window procedure result in messages being sent to the MLE. When the user makes a menu selection (e.g., selects Paste from the Edit menu) or uses one of the dialog boxes (e.g., to initiate a text search), a considerable amount of MLE message traffic is also generated — these will likewise be described. To gain the most from this discussion, I suggest that you follow along in the code as each issue is discussed (reading the code of Petzold, Duncan, and other pioneering OS/2 authors has helped me immensely).

Main()

As in all PM programs, this function contains three things: initialization code, the message loop, and termination code. During initialization, an anchor block (whatever that is) is obtained, a message queue is created, a window class is registered, and a standard window is created — this is all the usual “stuff”. For IPF (help), a HELPINIT structure is filled with information and a help instance is created. Finally, a PM timer is created — this will post a message to the client window procedure every 200 milliseconds to tell the editor it is time to update the cursor position display.

The message loop is nested inside an “infinite” for loop — the purpose of which is to give the user the chance to abort an exit, or to save the current contents of the edit window to a file before exit.

During termination, everything that was previously created is destroyed — including the help instance and the timer.

ClientWndProc()

A window procedure is what gives a window its behavioral characteristics — i.e., it is what makes one window behave differently from another (e.g., a pushbutton control window has a different window procedure than a list box control window). The client window procedure (of this or any PM application) is what mainly determines how the application looks, feels, and acts. A client window procedure is a massive switch/case statement, with one case to handle each of the messages that the window must respond to. In the case of the WM_COMMAND message, a further (nested) switch/case handles each type of WM_COMMAND message. Later, I will explain how many of these messages are handled, and how they affect the MLE.

SetPtrArrow() & SetPtrWait()

These two functions are used to alter the appearance of the mouse pointer. If a long operation (such as a text search) is initiated, a call to SetPtrWait changes the mouse pointer to a timer symbol (an hourglass or a clock). After the time-consuming operation is finished, a call to SetPtrArrow changes the mouse pointer back to normal.

TabWndProc()

This function is used for subclassing. The MLE window procedure is supplied by PM, and determines how the MLE behaves. Since I didn't like the way the MLE handled tabs, I used subclassing to alter the normal behavior. With subclassing, all messages destined for the MLE are sent first to the TabWndProc, which traps only WM_CHAR messages consisting of the Tab key so that it can convert the Tab to an appropriate number of spaces (all other messages are sent on to the regular MLE window procedure).

FindDlgProc()

This dialog box prompts the user for a text string to search for. When the user selects OK, the search proceeds. To facilitate repeated searches, any previous search string is “suggested” as a starting point. The string is placed in a global variable — code in the

♦

DO YOU REALLY
THINK WE'D LET YOU
GO TO MANAGEMENT
TO GET MONEY FOR
AN OBJECT DBMS
WITHOUT GIVING YOU
AMMUNITION
ABOUT WHY IT
MAKES BRILLIANT
FINANCIAL SENSE?

♦

Tell your management to forget all this nonsense about how the production reality of object database management systems is still years away. Tell them that ONTOS already has some 400 customers worldwide.

Let them know that some of the most widely respected leaders in communications, manufacturing, financial services, health care, and government are experiencing dramatic productivity gains and increased competitiveness—this very day—thanks to the inherent benefits of the ONTOS DB.

For a copy of our brochure, *The Impact of ONTOS*, phone 1-800-388-7110, Ext. 207. And when management asks you what the ONTOS Object DBMS is going to do for the enterprise, tell them...well, tell them to fire up their calculators.



© 1992, ONTOS, Inc., Three Burlington Woods, Burlington, MA 01803. All rights reserved.



AN OS/2-PM EDITOR USING MLE

client window procedure does the actual searching (via messages to the MLE).

ReplaceDlgProc()

Similar to above, this dialog box first prompts the user for a search string, then a replacement string. Options are provided to allow the user to continue the search without replacing the current occurrence, to replace just the current occurrence, or to replace all occurrences. As above, the actual work (search/replace) is carried out by code in the client window procedure (this will be covered later when messages are discussed).

GoLnDlgProc()

Prompts the user for a line number. When the user selects OK, that line number is displayed at the top of the edit window. Again, the dialog box procedure just collects user input — the client window procedure does the real work.

ReadFile()

Using a filename provided by OpenDlgProc, this function reads a file of up to 64K into a dynamically allocated buffer. Returns file size (or a negative code indicating an error); passes a pointer to the buffer back to the client window procedure, which transfers the data to the edit window (MLE).

MakeWriteBuffer() & WriteFile()

These two functions together are the mirror image of ReadFile — allocation of buffer space and the actual writing to the file must be split, due to the requirements of the MLE. The sequence of events is roughly: 1) ask MLE how much data it has; 2) allocate enough space for that data (that is what MakeWriteBuffer does); 3) transfer the data from the MLE to the buffer; and 4) create the file and write to it (that is what WriteFile does).

Release()

This function will release (i.e., free) the dynamically-allocated storage used to read or write files.

Message Traffic

Traditional applications are procedure-based — they use a programmed sequence of operations. The programmer of such traditional applications determine what the pattern of user interaction will be via hard coding. OS/2 Presentation Manager applications are very different — the user is more in control and may perform operations in any order. This focus on the user requires a different programming paradigm — the application must be ready to accept any command from the user at almost any time. This is accomplished by way of messages — when the user requests an operation, the application is advised by message. The application must

**Need more flexibility at the OS/2 prompt? More power in your batch files?
Then you need...**



JP Software's Enhanced Command Processor for OS/2

**32-bit version
now available!**

*4OS2 replaces the OS/2 command processor (CMD.EXE) with
a complete new interface, fully compatible with existing OS/2 commands.
It makes the [C:\] prompt friendly, powerful, and easy to use.*

- Full featured command line editing, command history, and recall.
- Over 40 new commands to make the command line a truly productive tool.
- Dozens of powerful new batch commands and functions, plus faster batch execution.
- Only \$89 bundled with 4DOS, JP Software's award-winning command processor for DOS (4OS2 only, \$69; \$29 for current 4DOS users).
- Aliases to create new or shorthand commands for common operations.
- Significant enhancements to most standard OS/2 commands.
- Complete, cross-referenced, on-line help.
- Shareware copies available on CompuServe (GO JPSOFT, library 10/JP Software) and many BBSes.
- Fully compatible with 4DOS and CMD.EXE, and with OS/2 versions 1.2, 1.3, and 2.0.

To order, call 800-368-8777 (order line), 617-646-3975, or 617-646-0904 (fax)

JPsoftware

P.O. Box 1470, East Arlington, MA 02174, USA

4OS2 is a trademark and 4DOS ® is a registered trademark of JP Software Inc. OS/2 ® is a registered trademark of IBM Corporation. Other company and product names are trademarks of their respective owners. Copyright ©1993, JP Software Inc.

be programmed to respond to any message that the user can generate. Specific messages are generated any time the user moves the mouse, strikes a key on the keyboard, or selects a menu entry (among other actions).

Most messages are dispatched to the client window procedure; when a dialog box is being displayed, the messages go to the dialog window procedure. Each of the messages that the Edit client window procedure is programmed to respond to is described below. In many cases, an incoming message (from the user) results in other messages being sent to the MLE — these also are described.

WM_CREATE

The first message received by the client window procedure is WM_CREATE, which is sent during processing of the WinCreateStdWindow function. The purpose of this message is to allow the window procedure to do any initialization prior to it being displayed. In the case of Edit, there is much to be done. The single most important job is to create the MLE window by calling WinCreateWindow with the window class set to WC_MLE — which causes the built-in (PM supplied) MLE window procedure to control this window. The MLE window is immediately subclassed to allow for special handling of tabs. To subclass a window, you must provide a new window procedure for this window. A pointer to the original window procedure is returned as a result of the subclassing operation — this function pointer is used (by TabWndProc) to call the original window

procedure from within the new window procedure to allow the original window procedure to still do most of its work in the usual way (e.g., in Edit, subclassing affects only the way that the MLE handles tabs). Other important actions taken during WM_CREATE message processing are the setting of various MLE options (font — set to monospace; maximum size — set to 64K; tabstops — set to 64 pels). Also, a few minor jobs are done during WM_CREATE processing — getting handles to various windows (required for later message processing) and determining the menu height (used for the height of the status line that displays the cursor position).

WM_SIZE

When the MLE is created, its position and size are not specified. However, during the WM_SIZE message the MLE is positioned and sized to just about fill the entire client area — less a strip along the top of the window (just below the menu) where cursor position (line and column) will be displayed. The WM_SIZE message is sent by PM every time the main (client) window changes size.

WM_TIMER

When the client window procedure receives a WM_TIMER message (about every 200 milliseconds), it sends a couple of messages to the MLE, and does a few calculations to determine the position (line and column) of the cursor. If the cursor position has changed, the client window is invalidated to cause PM to send a WM_PAINT

Worried about testing your OS/2, networked applications? Not if you're using the Softbridge Automated Test Facility.

If you're building OS/2 (or Windows) applications, you know that software testing is half the battle in the development cycle. If your applications are networked, they're even more complicated to test.

The Softbridge Automated Test Facility (ATF) was designed for unattended, centralized testing of OS/2 and Windows applications, networked or stand-alone.

ATF is used by corporations and software vendors who want to:

- ✓ **Build higher quality software**
- ✓ **Decrease development time & costs**
- ✓ **Make the impossible possible**

Don't just take our word for it ...

Cooperative Solutions used ATF to test Ellipse, and "Ellipse has been practically bug free." (*Datamation* - March 15, 1992)

"Reuters Information Service has used ATF for three months to test a client/server application. So far, ATF has enabled the firm to reduce the time required to perform 800 test cases from one month to one week." (*Network World* - June 22, 1992)

"Bachman Information Systems chose ATF because it's 70-80% better than anything on the market." (*PC Week* - June 15, 1992)



Softbridge

If software quality is on your checklist, call 800-955-9190 and learn more about ATF.

Softbridge, Inc. 125 CambridgePark Drive Cambridge, MA 02140 617-576-2257

AN OS/2-PM EDITOR USING MLE

message (so that the new cursor information is displayed).

WM_PAINT

During processing of a WM_PAINT message, the current cursor position (line and column) are displayed along the top of the client area just above the MLE window. No other client area drawing is needed because the MLE window procedure takes care of keeping the text properly displayed as the user enters or deletes text, or scrolls through the text, etc.

WM_MINMAXFRAME

When Edit has a file loaded into the MLE, the filename is displayed in the program's title bar. When the program is minimized (i.e., displayed as an icon) the title bar text is usually displayed under the icon. Unfortunately, PM limits the amount of text that can be displayed under an icon — a full filename (especially if the path is included) will likely be cut off. Each time the program is changed to or from an icon, PM sends a WM_MINMAXFRAME message — Edit uses this message to change the title bar text to just the program name when the program is minimized and back to the program name + the filename otherwise.

WM_INITMENU

Several of the program's features are not always available — for instance Cut, Copy, and Delete are not available unless some text has

been selected (highlighted) in the edit window. Similarly, Paste is not available unless the clipboard contains text. Each time PM is about to display a menu, it sends the program a WM_INITMENU message. When this message is received, Edit asks the clipboard if there is any text present and asks the MLE if any text has been highlighted. If not, the appropriate menu item is disabled (greyed out) so that the user cannot select an operation that is not currently valid.

WM_CONTROL

There are several messages that the MLE can send to the client window procedure — all are sent via notification codes as part of a WM_CONTROL message. Edit handles only two of these: MLN_OVERFLOW (the MLE has exceeded its 64K limit), and MLN_CHANGE (the text in the MLE has changed). The first of these is used to warn the user (who should then split the file before continuing). The second message (MLN_CHANGE) helps to determine if the file should be saved before exit and also helps to determine whether an undo operation has occurred (and can thus be redone by invoking undo again).

WM_COMMAND

Whenever the user selects a menu entry (whether by mouse or by keyboard), PM generates a WM_COMMAND message. Along with the WM_COMMAND message comes a parameter that indicates the nature of the command (i.e., which menu was selected). Just as the

PERFORMANCE 2.0

"Go as fast as 32 bits will take you"

A Tuning Kit for OS/2 2.0

PERFORMANCE 2.0 is both an instructional **booklet** & **diskette** with program objects that help you tune your system.

THE BOOKLET -60 pages, **best source for OS/2 Tuning Information.**

BASIC section clearly describes a 32 bit, preemptive multitasking, multi-threading, demand paging virtual memory system, *written for the novice!*

DETAIL section focuses on tip, techniques, and parameters explaining HPFS, FAT, Caching, Buffering, Swapping, Fragmentation, Multi threading, Timeslicing, and more - *written for the advanced user!*

THE DISKETTE - 25+ software OBJECTS - REXX source code included. Designed to help you tune your system with SIMPLICITY.

ENDORSEMENTS INCLUDE -

"...booklet alone worth the price of the package" - Bill Zack, Author OS/2 Handbook
"...all the tuning information in one place ...Highly Recommended!" - David Barnes, IBM Instructor
"...you won't be disappointed" - Satisfied Customer Quote on CompuServe
"...immediate performance improvement, immensely underpriced" - Customer Testimonial

***** Accredited by IBM I.V. LEAGUE *****
***** Recipient of IBM's OS/2 Software Developers AWARD *****

All for \$19.95 + 2.50 S&H
send Check or M.O. (\$22.45)
VISA & MC Accepted
or call

CLEAR & SIMPLE, INC
P.O. Box 130
W. Simsbury, CT 06092
(203) 658-1204

CLEAR & SIMPLE has obtained marketing rights for:

CONDUIT - A LAN File Distribution Package - LAN Independent- Novell, IBM, Banyan, etc
Automatically distributes current files and/or software from Server to Workstation.
Simple design - Profile driven - Security & Audit Trail - Automated or User Initiated
Pre/Post Download processing supported - Inventory Data Base - OS/2 & DOS

Introductory Price of only \$99/Server License - (Normal List \$299)

Realize the true vision of OS/2's
Workplace Shell

with...

ICONtroll!

Icon Manager and OS/2 Icon Library

A breakthrough operating system like OS/2 deserves a breakthrough icon manager and library. After all, you're not taking full advantage of object-orientation if the most important objects - YOUR programs and data files - all look alike.

Now you can easily assign the icons of your choice to related programs and data files in a SINGLE procedure, and let ICONtroll! do the busy work for you! And, ICONtroll! works AUTOMATICALLY to keep your desktop icons up-to-date as new data files are added to your system!

ICONtroll! includes the finest, most comprehensive library of OS/2 icon files ever assembled - over 300 dazzling images - in both large AND small formats. And, with the unique "Look-before-you-load" feature you don't have to load the entire library to use it - saving you time and disk space.

INTRODUCTORY OFFER!

ONLY \$59.95

(PLUS \$7.00 SHIPPING AND HANDLING)

ORANGE HILL SOFTWARE

P.O. BOX 907

ATWOOD, CA 92601-0907

(714) 524-5851

many messages themselves are handled by a large switch/case statement, the many menu commands are similarly handled by a (now nested) switch/case statement with one case for each menu selection. The following commands are processed:

IDM_NEW

The IDM_NEW command is the result of the user choosing New from the File menu (clears the edit window). If the edit window has changed, the user is given a chance to save to a file before this command is acted upon. The New operation is performed by first asking the MLE how much text it has, and then deleting it all.

IDM_OPEN

The IDM_OPEN command is the result of the user choosing the Open from the File menu (allows the user to open a new file). Edit first sends itself a WM_COMMAND:IDM_NEW message to clear the edit window in preparation. Next, the OpenDlgProc is invoked indirectly by calling WinDlgBox — if the user selects OK with a valid filename chosen, this procedure returns TRUE and the file is read by calling ReadFile. The buffer filled up by ReadFile is displayed in the edit window by sending the MLE several messages (after checking to be sure ReadFile was successful). The technique of “sending yourself messages”, as in the case of sending the WM_COMMAND:IDM_NEW message above, is a powerful alternative to using subroutines to get work done inside a window procedure.

IDM_SAVE

The IDM_SAVE command is the result of the user choosing Save from the File menu — using the technique just described, an IDM_SAVE command can also result (indirectly) from the user choosing Save as from the File menu. Processing the save command consists of asking the MLE how much text it has, making a write buffer for the text, transferring the text from the MLE to the buffer and finally writing the buffer to a file. In what almost seems like mutual recursion, the WM_COMMAND:IDM_SAVEAS message is sent if the edit window does not already have a name.

IDM_SAVEAS

The IDM_SAVEAS command is the result of the user choosing Save as from the File menu. The SaveasDlgProc is invoked (indirectly, by WinDlgBox) to allow the user to choose a filename. Once a filename is chosen, a boolean (hasName) is set to TRUE, and the WM_COMMAND:IDM_SAVE message is sent. It is this boolean which prevents any recursion resulting from IDM_SAVE invoking IDM_SAVEAS and vice versa.

IDM_EXIT

The IDM_EXIT command is the result of the user choosing Exit from the File menu. In turn, Edit sends a WM_CLOSE message, which eventually results in a WM_QUIT message being sent by PM. When the message loop receives a WM_QUIT message, the loop terminates. There are (at least) two other ways that the WM_CLOSE/WM_QUIT sequence can occur: as a result of the user choosing Close from the system menu, or End task from the PM Task List. In all cases the “infinite” for loop in main (described earlier)

gives the user a chance to save the contents of the edit window to a file before the program actually exits, or to cancel the shutdown.

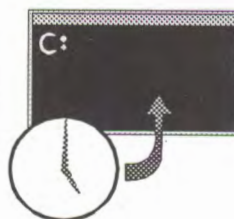
IDM_UNDO

The IDM_UNDO command is the result of the user choosing Undo from the Edit menu. The MLE knows how to undo the last operation, so all that Edit has to do is send the MLE an MLM_UNDO message. If the last action was undo, the undo is undone (i.e., redo). The undone flag is set to TRUE to allow redo (whether undo is enabled is controlled by the action of the WM_INITMENU command processing, explained above).

IDM_CUT, IDM_COPY, & IDM_DELETE

The IDM_CUT, IDM_COPY, and IDM_DELETE messages are all handled similarly. They are the result of the user choosing Cut, Copy, or Delete from the Edit menu. Since the MLE “knows how” to do all of these operations, all that is necessary is to send the appropriate message (MLM_CUT, MLM_COPY, or MLM_CLEAR) to the MLE — it does the rest. As mentioned above, these commands would have been disabled during WM_INITMENU processing if the user had not previously selected any text. The copy command places the selected text on the system clipboard; the delete command erases the selected text; the cut command does both (copies text to the clipboard and then erases it).

Schedule Programs & Reminders Automatically...



Chron v3.0
Event Scheduler for OS/2

Only \$69

Site Licensing Available

Uses:

- Schedule backups and database maintenance
- Remind yourself of recurring meetings
- Schedule long running or resource consumptive tasks for after hours



Hilbert Computing
1022 N. Cooper
Olathe, KS 66061

Voice: (913) 780-5051
BBS: (913) 829-2450
CIS: 73457,365

Both
16 & 32-bit
code
included!

AN OS/2-PM EDITOR USING MLE

IDM_PASTE

The IDM_PASTE command results from the user choosing Paste from the Edit menu. The MLE also "knows-how" to paste from the clipboard (all we have to do is send it an MLM_PASTE message). Similar to the case with cut, copy, and delete, the Paste menu selection would have been disabled during the WM_INITMENU processing if there were no text on the clipboard.

IDM_FIND

The IDM_FIND command results from the user choosing Find from the Edit menu. First, the FindDlgProc is invoked via a call to WinDlgBox — if the user does not select Cancel, WinDlgBox returns DID_OK and the search begins. Although the MLE "knows how" to search, a little more work is required: a structure must be filled with information before the MLM_SEARCH message is sent to the MLE. If the target string is found, the MLE highlights it. If the target is not found, the MLE does nothing visible, but it returns FALSE, which Edit uses to display a "not found" message box.

IDM_REPLACE

The IDM_REPLACE command results from the user choosing Replace from the Edit menu. The processing is somewhat similar to Find, except that two strings are involved (a target and a replacement) and there is a loop to allow repeated find/replace cycles. There are also options to replace either a single instance or all instances of the target with the replacement string. The first time the ReplaceDlgProc is called (via WinDlgBox), a parameter passed to the dialog box procedure causes the Replace and Replace All buttons to be disabled. For successive iterations, these buttons are enabled by altering the parameter mentioned earlier (i.e., setting "first" to FALSE). If, after the target is found, the user then chooses replace, an MLM_INSERT message is used to replace the single instance; if the user chooses replace all, MLM_INSERT replaces the first instance (the one that was already found), and an MLM_SEARCH message with a parameter of MLM_SEARCH_CHANGEALL is sent to find and replace the rest.

IDM_GO

The IDM_GO command results from the user choosing Go to line from the Edit menu. The processing is more complicated than it should be, because the MLE offers no way to go directly to a specific line number — you must go through two steps: first convert the line number to an absolute character position, and then cause the MLE to display that character on the first line of the edit window. Matters are complicated even further, because if the user then depresses a cursor movement key, the MLE will "snap back" to where it was prior to repositioning. The "trick" to avoiding this "snapping back" is to simulate a mouse button hit, which "locks" the MLE in its new position.

IDM_HELPFORHELP, IDM_EXTENDEDHELP, IDM_KEYSHelp, & IDM_HELPINDEX

The way that I handle processing of the help menu messages is contrary to the method recommended by IBM — however, my method gives me greater flexibility and is somewhat simpler and more consistent. In the resource file (EDIT.RC), IBM recommends that the help menu items (except for Help for help) generate predefined messages types by sending a WM_SYSCOMMAND message with a specific pa-

rameter (which is unlike other menus, and prevents the context-sensitive help that is available for other menus). Instead, I set up the help menu in the resource file in a manner that is consistent with other menus, and then send the WM_SYSCOMMAND messages from the client window procedure (for extended help, keys help, and the help index), which allows for context-sensitive help within the help menu. I also simplify the help table and help subtable by using the same constants for help as for the menus (e.g., HELPSUBITEM IDM_FILE, IDM_FILE) — this is much simpler than the recommended method (of relating the menu ID to some new ID), and causes no limitations or other problems. Most of the effort in putting together on-line help involves the IPF file, which is discussed below.

WM_ARGS & WM_CLEANFILE

Edit also uses two so-called user messages — these are messages defined by the application instead of by PM. All messages are identified by a unique number; any message defined by the application should be numbered starting at WM_USER (a pre-defined constant guaranteed to be above the message numbers used by PM). The first user message is WM_ARGS, which is actually sent from main (just before the message loop is entered) as a result of the user entering a filename on the command line when Edit is started. When the client window procedure receives this message, it opens up a file and reads it into the edit window (of the MLE). The other user message, WM_CLEANFILE, is sent whenever the editor is in a condition where it is safe to exit without saving the current edit window to a file. This message also causes the MLE undo-state to be reset. Edit sends itself the WM_CLEANFILE message whenever a new file is read, and whenever the edit window is saved to a file (these operations cannot be undone, but no need to save before exit).

HM_ERROR & HM_QUERY_KEYS_HELP

The application receives two messages related directly to the help manager; these are HM_ERROR (which results in the display of a message box informing the user that help is not available), and HM_QUERY_KEYS_HELP (which is merely passed back to PM for processing).

WM_DESTROY

After the user has terminated the application, PM sends a WM_DESTROY message — the only thing left to be done by Edit is to destroy the MLE (which was created during the WM_CREATE message).

Conclusion

Many people have lamented the long, steep learning curve that must be climbed to become productive programming in a GUI environment such as OS/2 Presentation Manager. This project illustrates that the powerful features provided by the GUI can make an otherwise very challenging project almost a snap. Without the MLE (Multi Line Edit predefined control window), the editor presented here would have taken considerably longer to develop and would have resulted in a considerably greater number of total lines of code. I submit that once mastered, a GUI such as the OS/2 Presentation Manager actually makes a programmer more productive and results in more functional and useful applications. These advantages far outweigh the disadvantage of any initial difficulty in learning the GUI paradigm. ♦


```

/* main window procedure for application */
MRESULT EXPENTRY ClientWndProc (HWND hwnd, USHORT msg, MPARAM mp1,
MPARAM mp2)
{
    BOOL first; /* set to TRUE if replace dialog called for first time */
    CHAR str[60]; /* character string for output messages */
    FATTRS fattr; /* font attributes -- try to change to monospaced */
    FONTMETRICS *pfm; /* find a monospaced system font */
    HDC hdc; /* device context */
    HPS hps; /* presentation space */
    LONG buflen; /* number of characters read into buffer */
    LONG goline, gochar; /* go to line specified by user */
    LONG lHRes, lVRes; /* device resolution */
    LONG lNumberFonts, lRequestFonts; /* look for 10 point font */
    LONG selmin, selmax; /* text selected for deletion */
    MLE_SEARCHDATA SearchData; /* used is MLM_SEARCH message */
    int i; /* simple loop counter */
    PCHAR buffer; /* buffer for file I/O */
    POINTS ps; /* simulate mouse click to place cursor on specified line */
    PSWP pswp; /* set window position structure */
    RECTL rcl; /* rectangle drawing coordinates */
    SHORT undoable; /* TRUE if last operation can be undone */
    static BOOL undone = FALSE; /* allows for redo immediately after undo */

    static HWND hwndFrame, hwndClient, hwndMenu, hwndMLE; /* window handles */

    static USHORT cyClient, cxClient; /* size of edit window */
    USHORT newline, newcolumn; /* latest cursor position -- changed? */
    static USHORT line, column; /* current cursor position */
    static USHORT menuheight; /* height of standard window */
    USHORT res; /* response from dialog or message box */
    USHORT resInfo; /* flags for query clipboard information */
    static USHORT resNew; /* result of IDM_NEW (allows Cancel) */

    switch (msg) {
        case WM_CREATE:
            hwndClient = hwnd;
            hwndFrame = WinQueryWindow (hwnd, QW_PARENT, FALSE);
            hwndMenu = WinWindowFromID (hwndFrame, FID_MENU);
            menuheight = (USHORT) WinQuerySysValue (HWND_DESKTOP, SV_CYMENU);
            WinSetWindowText (hwndFrame, "EDIT");

            hwndMLE = WinCreateWindow (
                hwndClient,
                WC_MLE,
                "",
                WS_VISIBLE | MLS_HSCROLL | MLS_VSCROLL | MLS_BORDER,
                0, 0, 0, 0, /* will set size & position later */
                hwndClient,
                HWND_TOP,
                ID_MLE,
                NULL, NULL);

            /* subclass to intercept tabs (convert to spaces) */
            pfMLE = WinSubclassWindow (hwndMLE, TabWndProc);

            /* override v2.x MLE colors */
            WinSendMsg (hwndMLE, MLM_SETTEXTCOLOR,
                MPFROMLONG (CLR_DEFAULT), 0L);
            WinSendMsg (hwndMLE, MLM_SETBACKCOLOR,
                MPFROMLONG (CLR_BACKGROUND), 0L);

            /* try to switch to System Monospaced 10-point font */
            hps = WinGetPS (hwndMLE);
            hdc = GpiQueryDevice (hps);
            DevQueryCaps (hdc, CAPS_HORIZONTAL_FONT_RES, 1L, &lHRes);
            DevQueryCaps (hdc, CAPS_VERTICAL_FONT_RES, 1L, &lVRes);
            lRequestFonts = 0L;
            lNumberFonts = GpiQueryFonts (hps, QF_PUBLIC, "System
Monospaced",
                &lRequestFonts, 0L, NULL);
            pfm = malloc ((SHORT) lNumberFonts * sizeof (FONTMETRICS));
            GpiQueryFonts (hps, QF_PUBLIC, "System Monospaced",
                &lNumberFonts, (LONG) sizeof (FONTMETRICS), pfm);

            for (i = 0; i < (int)lNumberFonts; i++) {
                if (pfm[i].sDeviceRes == (SHORT)lHRes && /* does font... */

                pfm[i].sDeviceRes == (SHORT)lVRes && /* match device? */

                pfm[i].sNominalPointSize == 100) { /* 10 point? */
                    WinSendMsg (hwndMLE, MLM_QUERYFONT,
                        MPFROMP (&fattr), (MPARAM) 0L);
                    fattr.lMatch = pfm[i].lMatch;
                    strcpy (fattr.szFacedName, "System Monospaced");
                    WinSendMsg (hwndMLE, MLM_SETFONT,
                        MPFROMP (&fattr), (MPARAM) 0L);
                    break; /* exit for loop */
                }
            }
            free (pfm);
            WinReleasePS (hps);

            /* set up some MLE parameters */
            WinSendMsg (hwndMLE, MLM_SETTEXTLIMIT,
                MPFROMLONG (65535L), (MPARAM) 0L);

            WinSendMsg (hwndMLE, MLM_SETTABSTOP,
                MPFROMSHORT (64), (MPARAM) 0L);

            WinSendMsg (hwndMLE, MLM_FORMAT,
                MPFROMSHORT (MLFIE_CFTXT), (MPARAM) 0L);

            WinSendMsg (hwndMLE, MLM_RESETUNDO, (MPARAM) 0L, (MPARAM) 0L);
            WinSetFocus (HWND_DESKTOP, hwndMLE);
            return 0;

        case WM_MINMAXFRAME:
            pswp = PVOIDFROMMP (mp1);
            if (pswp->fs & SWP_MINIMIZE)
                WinSetWindowText (hwndFrame, "EDIT");
            else {
                if (hasName) {
                    sprintf (str, "EDIT -- %s", szFileName);
                    WinSetWindowText (hwndFrame, str);
                }
            }
            return 0;

        case WM_INITMENU:
            if (SHORT1FROMMP (mp1) == IDM_EDIT) {
                /* enable Cut, Copy, or Delete only if text selected */
                selmin = LONGFROMMR (WinSendMsg (hwndMLE, MLM_QUERYSEL,
                    (MPARAM) MLFQS_MINSEL, (MPARAM) 0L));
                selmax = LONGFROMMR (WinSendMsg (hwndMLE, MLM_QUERYSEL,
                    (MPARAM) MLFQS_MAXSEL, (MPARAM) 0L));

                WinSendMsg (hwndMenu, MM_SETITEMATTR,
                    MPFROM2SHORT (IDM_CUT, TRUE),
                    MPFROM2SHORT (MIA_DISABLED,
                        (selmin == selmax) ? MIA_DISABLED : 0));
                WinSendMsg (hwndMenu, MM_SETITEMATTR,
                    MPFROM2SHORT (IDM_COPY, TRUE),
                    MPFROM2SHORT (MIA_DISABLED,
                        (selmin == selmax) ? MIA_DISABLED : 0));
                WinSendMsg (hwndMenu, MM_SETITEMATTR,
                    MPFROM2SHORT (IDM_DELETE, TRUE),
                    MPFROM2SHORT (MIA_DISABLED,
                        (selmin == selmax) ? MIA_DISABLED : 0));

                /* enable Paste only if data available in Clipboard */
                WinSendMsg (hwndMenu, MM_SETITEMATTR,
                    MPFROM2SHORT (IDM_PASTE, TRUE),
                    MPFROM2SHORT (MIA_DISABLED,
                        WinQueryClipbrdFmtInfo (hab, CF_TEXT, &usfInfo)
                        ? 0 : MIA_DISABLED));

                /* enable Undo only if operation may be undone */
                undoable = SHORT1FROMMR (WinSendMsg (hwndMLE, MLM_QUERYUNDO,
                    (MPARAM) 0L, (MPARAM) 0L));
                WinSendMsg (hwndMenu, MM_SETITEMATTR,
                    MPFROM2SHORT (IDM_UNDO, TRUE),
                    MPFROM2SHORT (MIA_DISABLED, (undoable || undone)
                        ? 0 : MIA_DISABLED));
            }
            return 0;
    }
}

```



```

case WM_TIMER:
    if (SHORT1FROMMP (mp1) == ID_UPDATE) {
        /* determine position (line/column) of text cursor (caret) */
        selmin = LONGFROMMMR (WinSendMsg (hwndMLE, MLM_QUERYSEL,
            (MPARAM) MLFQS_MINSEL, (MPARAM) 0L));
        newline = (int) LONGFROMMMR (WinSendMsg (hwndMLE,
            MLM_LINEFROMCHAR,
            MPFROMLONG (selmin),
            (MPARAM) 0L));
        newcolumn = (int) (selmin - LONGFROMMMR (WinSendMsg (hwndMLE,
            MLM_CHARFROMLINE,
            MPFROMLONG ((long) newline),
            (MPARAM) 0L)));

        /* update on screen only if line or column changed */
        if (newline != line || newcolumn != column) {
            line = newline;    column = newcolumn;
            WinInvalidateRect (hwnd, NULL, FALSE);
        }
    }
    return 0;

case WM_CONTROL:
    switch (SHORT2FROMMP (mp1)) {
        case MLN_OVERFLOW:
            if (SHORT1FROMMP (mp1) == ID_MLE) {
                WinMessageBox (HWND_DESKTOP, hwnd,
                    "File too large -- 64K limit exceeded.",
                    "Error", 0, MB_OK | MB_ICONHAND | MB_MOVEABLE);
            }
            return 0;

        case MLN_CHANGE:
            if (SHORT1FROMMP (mp1) == ID_MLE) {
                NeedToSave = TRUE;
                undone = FALSE;
            }
            return 0;

        default:
            break;
    }
    break;

case WM_CLEANFILE:
    WinSendMsg (hwndMLE, MLM_RESETUNDO, (MPARAM) 0L, (MPARAM) 0L);
    NeedToSave = FALSE;
    return 0;

case WM_ARGS:
    /* read file into buffer */
    bufferlen = ReadFile (szFileName, &buffer);
    if (bufferlen == CANTREAD) {
        WinMessageBox (HWND_DESKTOP, hwnd,
            "Specified file does not exist.",
            "New File", 0, MB_OK | MB_ICONASTERISK | MB_MOVEABLE);
        hasName = TRUE;
        sprintf (str, "EDIT -- %s", szFileName);
        WinSetWindowText (hwndFrame, str);
        NeedToSave = TRUE;
    }
    else if (bufferlen == TOOLONG) {
        WinMessageBox (HWND_DESKTOP, hwnd,
            "File too large -- 64K limit exceeded.",
            "Error", 0, MB_OK | MB_ICONHAND | MB_MOVEABLE);
        WinSetWindowText (hwndFrame, "EDIT");
    }
    else if (bufferlen == NOMEMORY) {
        WinMessageBox (HWND_DESKTOP, hwnd,
            "Cannot allocate memory.",
            "Error", 0, MB_OK | MB_ICONHAND | MB_MOVEABLE);
        WinSendMsg (hwnd, WM_QUIT, (MPARAM) 0L, (MPARAM) 0L);
    }
    else {
        /* normal */
        /* transfer buffer to MLE */
        WinSendMsg (hwndMLE, MLM_SETIMPORTEXP,

```

```

        MPFROMMP (buffer),
        MPFROMSHORT ((USHORT)bufferlen));
        selmin = 0L;
        WinSendMsg (hwndMLE, MLM_IMPORT,
            MPFROMMP (&selmin), MPFROMLONG (bufferlen));
        /* free buffer */
        Release (buffer);
        hasName = TRUE;
        sprintf (str, "EDIT -- %s", szFileName);
        WinSetWindowText (hwndFrame, str);
        WinPostMsg (hwnd, WM_CLEANFILE, (MPARAM) 0L, (MPARAM) 0L);
    }
    return 0;

case WM_COMMAND:
    switch (SHORT1FROMMP (mp1)) {
        case IDM_NEW:
            if (NeedToSave) {
                res = WinMessageBox (HWND_DESKTOP, hwnd,
                    "Save current file?", "New", 0,
                    MB_YESNOCANCEL | MB_ICONQUESTION |
                    MB_MOVEABLE);

                if (res == MBID_YES) {
                    WinSendMsg (hwnd, WM_COMMAND,
                        MPFROM2SHORT (IDM_SAVE, 0), (MPARAM) 0L);
                }
                else if (res == MBID_CANCEL) {
                    resNew = MBID_CANCEL;
                    return 0;
                }
            }
            SetPtrWait();

            /* empty MLE */
            selmin = 0L;
            selmax = LONGFROMMMR (WinSendMsg (hwndMLE,
                MLM_QUERYTEXTLENGTH,
                (MPARAM) 0L, (MPARAM) 0L));
            WinSendMsg (hwndMLE, MLM_DELETE,
                MPFROMLONG (selmin), MPFROMLONG (selmax));

            hasName = FALSE;
            WinSetWindowText (hwndFrame, "EDIT");
            WinPostMsg (hwnd, WM_CLEANFILE, (MPARAM) 0L, (MPARAM) 0L);

            SetPtrArrow();
            resNew = MBID_OK;
            return 0;

        case IDM_OPEN:
            WinSendMsg (hwnd, WM_COMMAND,
                MPFROMSHORT (IDM_NEW), (MPARAM) 0L);
            if (resNew == MBID_CANCEL)
                return 0;

            if (WinDlgBox (HWND_DESKTOP, hwnd, OpenDlgProc,
                0, IDD_OPEN, NULL)) {
                SetPtrWait();

                /* user selected a (valid) file to open */
                /* read file into buffer */
                bufferlen = ReadFile (szFileName, &buffer);

                SetPtrArrow();

                if (bufferlen == CANTREAD) {
                    WinMessageBox (HWND_DESKTOP, hwnd,
                        "Specified file does not exist.",
                        "New File", 0, MB_OK | MB_ICONASTERISK |
                        MB_MOVEABLE);
                    hasName = TRUE;
                    sprintf (str, "EDIT -- %s", szFileName);
                    WinSetWindowText (hwndFrame, str);
                    NeedToSave = TRUE;
                }
            }

```



```

else if (bufferlen == TOOLONG) {
    WinMessageBox (HWND_DESKTOP, hwnd,
        "File too large -- 64K limit exceeded.",
        "Error", 0, MB_OK | MB_ICONHAND | MB_MOVEABLE);
    WinSetWindowText (hwndFrame, "EDIT");
    NeedToSave = FALSE;
}
else if (bufferlen == NOMEMORY) {
    WinMessageBox (HWND_DESKTOP, hwnd,
        "Cannot allocate memory.",
        "Error", 0, MB_OK | MB_ICONHAND | MB_MOVEABLE);
    WinSendMsg (hwnd, WM_QUIT, (MPARAM) 0L, (MPARAM) 0L);
}
else { /* normal */
    SetPtrWait();

    /* transfer buffer to MLE */
    WinSendMsg (hwndMLE, MLM_SETIMPORTEXP,
        MPFROMP (buffer), MPFROMSHORT
        ((USHORT)bufferlen));
    selmin = 0L;
    WinSendMsg (hwndMLE, MLM_IMPORT,
        MPFROMP (&selmin), MPFROMLONG (bufferlen));
    /* free buffer */
    Release (buffer);
    hasName = TRUE;
    sprintf (str, "EDIT -- %s", szFileName);
    WinSetWindowText (hwndFrame, str);
    WinPostMsg (hwnd, WM_CLEANFILE, (MPARAM) 0L, (MPARAM)
0L);

    SetPtrArrow();
}
return 0;

case IDM_SAVE:
    if (hasName) {
        /* determine amount of text in MLE */
        bufferlen = LONGFROMMR (WinSendMsg (hwndMLE,
            MLM_QUERYFORMATTEXTLENGTH,
            (MPARAM) 0L, MPFROMLONG (-
1L)));

        /* allocate space for buffer */
        if (NOMEMORY == MakeWriteBuffer (bufferlen, &buffer)) {
            WinMessageBox (HWND_DESKTOP, hwnd,
                "Cannot allocate memory.",
                "Error", 0, MB_OK | MB_ICONHAND | MB_MOVEABLE);
            WinSendMsg (hwnd, WM_QUIT, (MPARAM) 0L, (MPARAM) 0L);
        }
        SetPtrWait();

        /* transfer text from MLE to buffer */
        WinSendMsg (hwndMLE, MLM_SETIMPORTEXP,
            MPFROMP (buffer),
            MPFROMSHORT ((USHORT)bufferlen));
        selmin = 0L; selmax = bufferlen;
        WinSendMsg (hwndMLE, MLM_EXPORT,
            MPFROMP (&selmin), MPFROMP (&selmax));

        /* write to file */
        if (CANTWRITE == WriteFile (szFileName, bufferlen,
buffer)) {
            WinMessageBox (HWND_DESKTOP, hwnd,
                "Unable to write to file.",
                "Error", 0, MB_OK | MB_ICONHAND | MB_MOVEABLE);
        }

        /* deallocate buffer */
        Release (buffer);
        WinPostMsg (hwnd, WM_CLEANFILE, (MPARAM) 0L, (MPARAM)
0L);

        SetPtrArrow();
    }
    else {
        WinSendMsg (hwnd, WM_COMMAND,
            MPFROM2SHORT (IDM_SAVEAS, 0), (MPARAM) 0L);
    }
    return 0;

case IDM_SAVEAS:
    if (WinDlgBox (HWND_DESKTOP, hwnd, SaveasDlgProc,
        0, IDD_SAVEAS, NULL)) {
        hasName = TRUE;
        sprintf (str, "EDIT -- %s", szFileName);
        WinSetWindowText (hwndFrame, str);
        WinSendMsg (hwnd, WM_COMMAND,
            MPFROM2SHORT (IDM_SAVE, 0), (MPARAM) 0L);
    }
    return 0;

case IDM_EXIT:
    WinSendMsg (hwnd, WM_CLOSE, (MPARAM) 0L, (MPARAM) 0L);
    return 0;

case IDM_UNDO:
    WinSendMsg (hwndMLE, MLM_UNDO, (MPARAM) 0L, (MPARAM) 0L);
    undone = TRUE;
    return 0;

case IDM_CUT:
    WinSendMsg (hwndMLE, MLM_CUT, (MPARAM) 0L, (MPARAM) 0L);
    return 0;

case IDM_COPY:
    WinSendMsg (hwndMLE, MLM_COPY, (MPARAM) 0L, (MPARAM) 0L);
    return 0;

case IDM_PASTE:
    WinSendMsg (hwndMLE, MLM_PASTE, (MPARAM) 0L, (MPARAM) 0L);
    return 0;

case IDM_DELETE:
    WinSendMsg (hwndMLE, MLM_CLEAR, (MPARAM) 0L, (MPARAM) 0L);
    return 0;

case IDM_FIND:
    if (DID_OK == WinDlgBox (HWND_DESKTOP, hwnd, FindDlgProc,
        0, IDD_FIND, NULL)) {
        SetPtrWait();

        SearchData.cb = sizeof (MLE_SEARCHDATA);
        SearchData.pchFind = szFind;
        SearchData.cchFind = strlen (szFind);
        SearchData.iptStart = (-1L);
        SearchData.iptStop = (-1L);

        res = SHORT1FROMMR (WinSendMsg (hwndMLE, MLM_SEARCH,
            MPFROMLONG
            (MLFSEARCH_SELECTMATCH),
            MPFROMP (&SearchData)));

        SetPtrArrow();

        if (!res) {
            WinMessageBox (HWND_DESKTOP, hwnd,
                "Search string not found.", "Find", 0,
                MB_OK | MB_ICONASTERISK | MB_MOVEABLE);
        }
        return 0;

case IDM_REPLACE:
    first = TRUE;
    res = WinDlgBox (HWND_DESKTOP, hwnd, ReplaceDlgProc,
        0, IDD_REPLACE, (PVOID)(&first));
    first = FALSE;
    for (;;) {
        if (res == DID_CANCEL)
            break;
        else if (res == DID_OK) {
            SetPtrWait();

```



```

        SearchData.cb = sizeof (MLE_SEARCHDATA);
        SearchData.pchFind = szFind;
        SearchData.cchFind = strlen (szFind);
        SearchData.iptStart = (-1L);
        SearchData.iptStop = (-1L);

        res = SHORT1FROMMMR (WinSendMessage (hwndMLE, MLM_SEARCH,
        MPFROMLONG
(MLFSEARCH_SELECTMATCH),
        MPFROMP (&SearchData)));

        SetPtrArrow();

        if (!res) {
            WinMessageBox (HWND_DESKTOP, hwnd,
                "Search string not found.", "Replace", 0,
                MB_OK | MB_ICONASTERISK | MB_MOVEABLE);
            break; /* exit search/replace */
        }
    }
    else if (res == DID_DOREPLACE)
        WinSendMessage (hwndMLE, MLM_INSERT,
            MPFROMP (szReplace), (MPARAM) 0L);
    else if (res == DID_REPLACEALL) {
        SetPtrWait();

        WinSendMessage (hwndMLE, MLM_INSERT,
            MPFROMP (szReplace), (MPARAM) 0L);

        SearchData.cb = sizeof (MLE_SEARCHDATA);
        SearchData.pchFind = szFind;
        SearchData.cchFind = strlen (szFind);
        SearchData.pchReplace = szReplace;
        SearchData.cchReplace = strlen (szReplace);
        SearchData.iptStart = (-1L);
        SearchData.iptStop = (-1L);
        WinSendMessage (hwndMLE, MLM_SEARCH,
            MPFROMLONG (MLFSEARCH_CHANGEALL),
            MPFROMP (&SearchData));

        SetPtrArrow();

        WinMessageBox (HWND_DESKTOP, hwnd,
            "All occurrences replaced.", "Replace", 0,
            MB_OK | MB_ICONASTERISK | MB_MOVEABLE);
        break; /* exit search/replace */
    }
    else
        break; /* exit search/replace */

    res = WinDlgBox (HWND_DESKTOP, hwnd, ReplaceDlgProc,
        0, IDD_REPLACE, (PVOID) (&first));
}
return 0;

case IDM_GO:
    res = WinDlgBox (HWND_DESKTOP, hwnd, GoLnDlgProc,
        0, IDD_GOLINE, NULL);
    if (res == MBID_OK) {
        if (sscanf (szLine, "%ld", &goline)) {
            if (goline < 1)
                goline = 1;
            gochar = LONGFROMMMR (WinSendMessage (hwndMLE,
                MLM_CHARFROMLINE,
                MPFROMLONG (--goline),
                (MPARAM) 0L));
            WinSendMessage (hwndMLE, MLM_SETFIRSTCHAR,
                MPFROMLONG (gochar), (MPARAM) 0L);
            ps.x = 0; ps.y = cyClient - (2 * menuheight);
            WinSendMessage (hwndMLE, WM_BUTTON1DOWN,
                MPFROM2SHORT (ps.x, ps.y), (MPARAM) 0L);
            WinSendMessage (hwndMLE, WM_BUTTON1UP,
                MPFROM2SHORT (ps.x, ps.y), (MPARAM) 0L);
        }
    }
    return 0;

```

```

        case IDM_HELPFORHELP:
            if (hwndHelpInstance)
                WinSendMessage (hwndHelpInstance, HM_DISPLAY_HELP, 0L, 0L);
            break;

        case IDM_EXTENDEDHELP:
            WinPostMsg (hwndFrame, WM_SYSCOMMAND,
                MPFROM2SHORT (SC_HELPTEXTENDED, 0), (MPARAM) 0L);
            break;

        case IDM_KEYSHELP:
            WinPostMsg (hwndFrame, WM_SYSCOMMAND,
                MPFROM2SHORT (SC_HELPKEYS, 0), (MPARAM) 0L);
            break;

        case IDM_HELPINDEX:
            WinPostMsg (hwndFrame, WM_SYSCOMMAND,
                MPFROM2SHORT (SC_HELPINDEX, 0), (MPARAM) 0L);
            break;

        case IDM_ABOUT:
            WinDlgBox (HWND_DESKTOP, hwnd, AboutDlgProc,
                0, IDD_ABOUT, NULL);
            return 0;

        default:
            break;
    }
    break;

case WM_SIZE:
    cxClient = SHORT1FROMMP (mp2);
    cyClient = SHORT2FROMMP (mp2);

    WinSetWindowPos (
        hwndMLE,
        HWND_TOP,
        0, 0,
        cxClient, cyClient - menuheight,
        SWP_MOVE | SWP_SIZE);
    WinSetFocus (HWND_DESKTOP, hwndMLE);
    return 0;

case WM_PAINT:
    hps = WinBeginPaint (hwnd, NULL, NULL);
    sprintf (str, " Line: %5d Col: %5d", line + 1, column + 1);
    WinQueryWindowRect (hwnd, &rcl);
    rcl.yBottom = cyClient - menuheight;
    WinDrawText (hps, -1, str, &rcl, CLR_NEUTRAL, CLR_BACKGROUND,
        DT_LEFT | DT_VCENTER | DT_ERASERECT);
    WinEndPaint (hps);
    return 0;

case HM_QUERY_KEYS_HELP:
    return ((MRESULT) IDH_KEYSHELP);
    break;

case HM_ERROR:
    if ((hwndHelpInstance && (ULONG) mp1) == HMERR_NO_MEMORY) {
        WinMessageBox (HWND_DESKTOP, HWND_DESKTOP,
            "Help Terminated Due to Error", "Help Error", 0,
            MB_OK | MB_ICONEXCLAMATION | MB_MOVEABLE);
        WinDestroyHelpInstance (hwndHelpInstance);
    }
    else {
        WinMessageBox (HWND_DESKTOP, HWND_DESKTOP,
            "Help Error Occurred", "Help Error", 0,
            MB_OK | MB_ICONEXCLAMATION | MB_MOVEABLE);
    }
    break;

case WM_DESTROY:
    WinDestroyWindow (hwndMLE);
    return 0;
}
return WinDefWindowProc (hwnd, msg, mp1, mp2);

```


PAINT BY NUMBERS

THE ULTIMATE OS/2 GAME

**TIMUR
TABI**

Starting this month, the changes to the program are described on a module-by-module basis. This new format will make following the progress of the game much easier. At the end of this article, there is an expanded list of definitions. Most of the new terms are concerned with the enhancements to the targeting features.

Module ABOUT:

This file has been merged into a new module, MENU (see below). The "About..." dialog box has been expanded to include the toll-free number for subscription information.

Module FILES:

This month presents a new feature in OS/2 2.0 — the file dialog box. The WinFileDialog function creates and

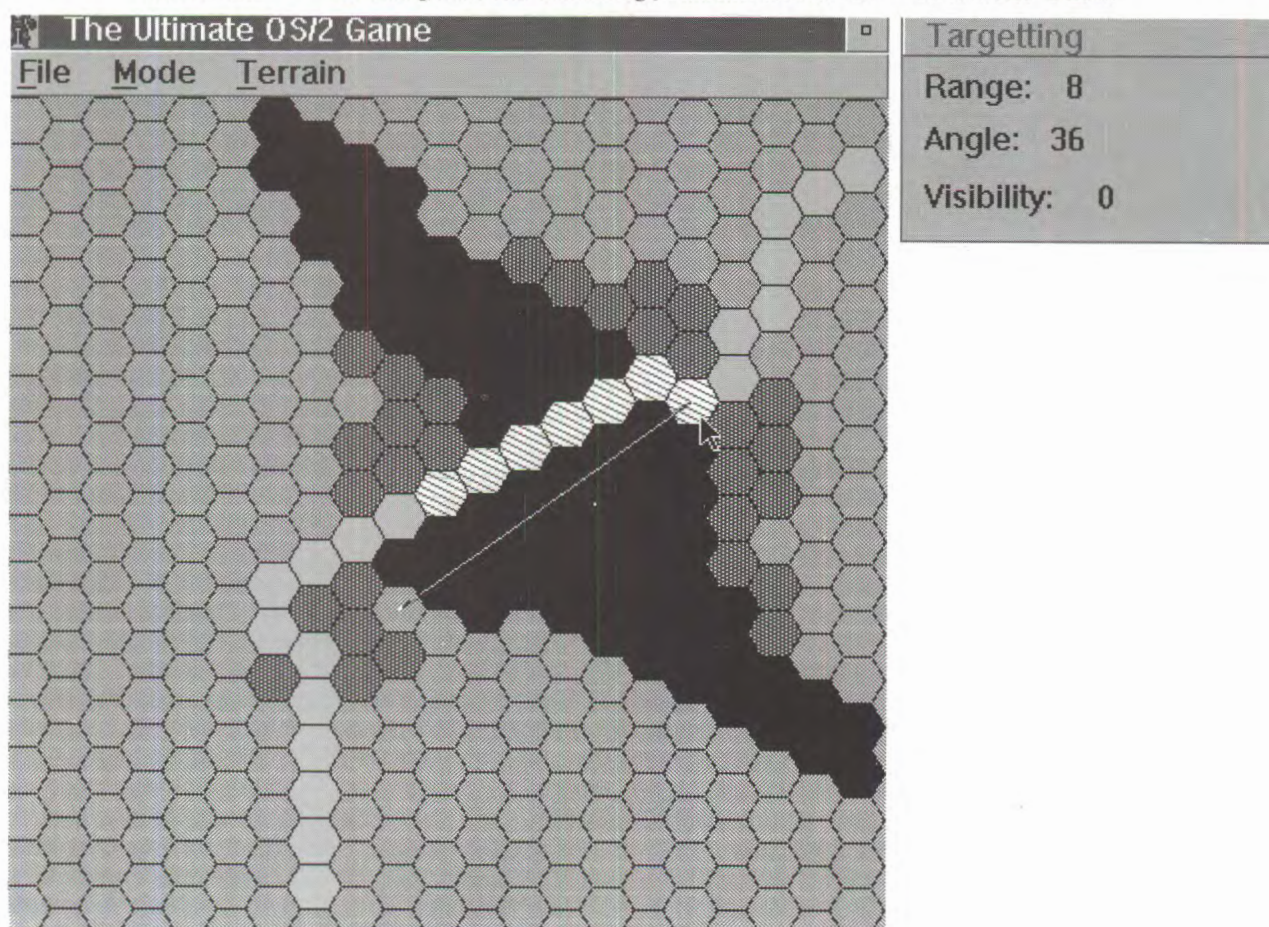
controls a CUA-compliant file selection dialog box. The operating system handles all the file and directory manipulations and returns the selected filename. As you might remember from the September issue, there is a lot of work involved in interpreting user input. This function should have been included long ago.

Module GAME:

The main source file for this project has some of the fat trimmed from it. The bulk of the code for the WM_PAINT, WM_MOUSEMOVE, and WM_COMMAND messages in WinProc() has been moved to other modules, where they belong.

Function main() loads and positions the info box, which is a small window that displays useful information during targeting. See the discussion of module TARGET below.

Timur Tabi can be reached at 2 Lloyd Haven Drive; Huntington, NY 11743. Source code is available on the OS/2 Shareware BBS, 703-385-4325.



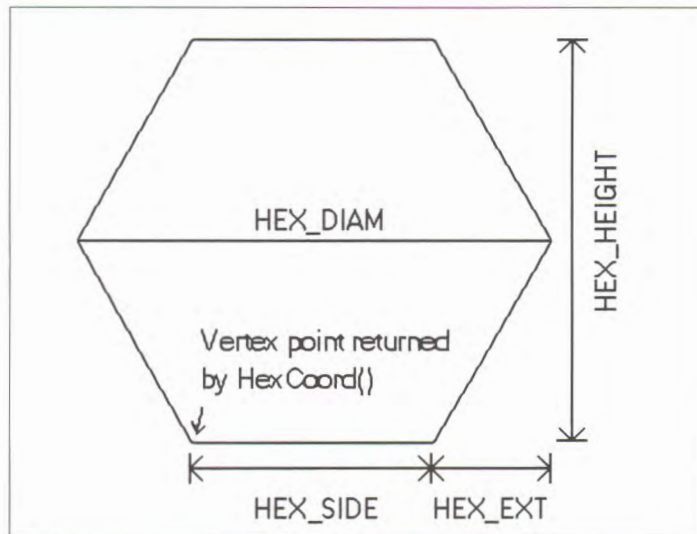


Figure 1

I recently received a letter from one of the technical editors of this magazine, informing me that the presentation space handles returned by WinGetPSO should not be kept open in global variables. This function creates a cached-micro presentation space, designed for fast, short-term drawing on the screen only. Although I have yet to witness any of the drawing errors that were described to me (such as drawing onto an overlapping window), an improved technique will nevertheless be incorporated. Look for it in next month's issue.

Module HEXES:

Figure 1 shows a hexagon labelled with various measurements. These values are currently constants, but they will be redefined as variables when the code for sizing and scaling the hex map is written. This enhancement won't be included until sometime next year.

Function HexLocate() is used to determine the hexagonal location of the mouse pointer when the mouse button is depressed. Instead of scanning the entire hexagonal grid until a match is made, the new version of this function makes an estimate of the pointer's location. It then tests the nearby hexagons by calling the improved HexInPoint(), which now checks the side triangles, as well as the inner rectangle.

The computer calculates the targeting path and displays it on the screen as you are targeting. Several new functions in this module are used to calculate the sequence of hexagons that compose this path. For each hexagon in the path, the targeting line enters at one side and exits at another. The targeting path is determined by finding these sides, one hexagon at a time.

Figure 2 shows the ordering of the quadrants and the inner geometry of a hexagon. Note that a vertex angle does not lie on any quadrant. The quadrant number is the same as the value returned by HexFirstSide() and HexNextSide().

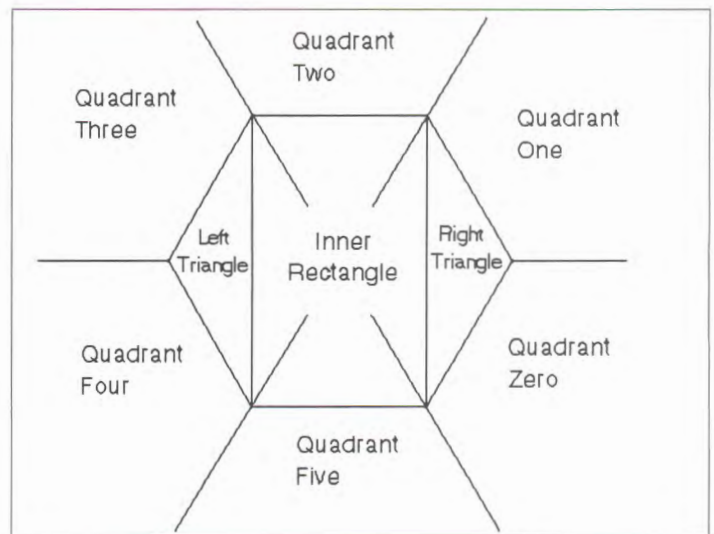


Figure 2

Field 'hiStart' in structure 'target' is the hex index of the source hexagon. Function HexFirstSide() calculates the side of the source hexagon from which the targeting line emerges. This gives you the second member of the targeting path. The computer knows the entry side for this hexagon. The targeting line must exit at one of the other five sides. Function HexPointFromSide() returns the x and y coordinates of the endpoints for a given side and hexagon. This is used to determine whether a particular side intersects the targeting line. Once the exit side is known, function HexFromSide() can tell you the index of the next hexagon. This sequence continues until the target hexagon (target.hiEnd) is reached. The other targeting path functions are contained in module TARGET, which is discussed later in this article.

Function HexDrawMap() draws the playing field. This function was originally contained within the WM_PAINT message-processing routine in module GAME. Note that GpiBox() is used instead of WinFillRect() to paint the client window, since WinFillRect() doesn't support patterns. The window is painted with the same color and pattern of the default terrain. Only the hexagons that are not the default terrain are drawn, resulting in a significant speed increase.

The performance can be further enhanced by drawing the black hexagonal grid in one shot, instead of one hexagon at a time. You might recall that GpiPolyLine() is used to draw all six sides of one hexagon. Instead of drawing each hex individually (which results in many overlapping line segments), the entire grid will be drawn with one call to GpiPolyLine(). This enhancement will be shown next month.

Module MENU:

This module is primarily used to house the MainCommand() function, which processes all the pull-down menu commands. The menus have changed slightly, but the code is mostly identical to that contained in the last article. Instead of toggling edit mode

(where "off" means that targeting mode is on), the user simply selects the desired mode. Additional modes will be available as the game develops.

Module TARGET:

There are two new features in the targeting module. The first is a status window (called the "info box") that shows the angle, range, and visibility of the target. The second is a visual representation of the targeting path.

The info box is actually a standard dialog box created with the Dialog Editor and displayed in `main()` with a call to `WinLoadDlg()`, which loads a modeless dialog box. This technique has the following advantages:

1. You can use the dialog editor to design the window.
2. A window procedure is not required! OS/2's default window procedure does all the work.
3. Only one function call is used to load and display the dialog box. Use another to position it on the screen. No other maintenance is required.
4. The text can be changed simply by sending messages. Just be sure you assign identifiers to blank text controls in the dialog box. Use these identifiers in the parameters for `WinSetDlgItemText()`. See function `TgtMove()` in module TARGET.

Functions `GetAngle()` and `GetRange()` calculate the targeting an-

gle and the range. Note that the Pythagorean theorem is not used to determine the range. The distance between two hexagons is defined as the length of the shortest path between them. A careful inspection of a hexagonal map reveals some interesting properties.

First, calculate `dx` and `dy`, the horizontal and vertical differences in the hex indices, respectively. Note that `dy` is always even, since the row indices for a given column are either all odd or all even. If `abs(dy)` is less than or equal to `abs(dx)` (i.e., the vertical component of the distance is not greater than the horizontal), then the range is always equal to just `dx`. Otherwise, the range is one-half the difference between `abs(dy)` and `abs(dx)`, added to `dx`.

For each movement of the targeting line, the path must be traversed three times - once for drawing the line, once for erasing it, and once for calculating the visibility. To save time, the slope and y-intercept of the targeting line are pre-calculated in function `TgtInitPath()`. These two variables are used by function `Intercept()` to determine whether a particular side of a particular hexagon intersects the targeting line. Function `NextHexSide()` uses `Intercept()` to locate where the targeting line exits a hexagon. With that information, function `HexFromSide()` in module HEXES can locate the next hexagon in the path.

There is one small problem with this approach. Experienced BattleTech players would probably claim that the algorithm is too accurate. It includes hexagons that the targeting line only skims. Technically speaking, these hexagons are part of the targeting path, since the line does pass through them. But in reality, a hexagon should only be included if it makes a significant contribution to the total visibility. A line of sight which touches only the edge of a hexagon should not include that hexagon in the targeting calculations.

How can this problem be solved? By determining whether the targeting line crosses near a vertex. If it does, then it is likely that the next hexagon will be crossed along the edges, resulting in the inclusion of an unwanted hexagon. Because of error propagation, the targeting path can include up to 25% too many hexagons.

First, the program should locate the nearby vertex. Then it should test the two sides of the hexagon that join at that vertex. One of these two sides is the correct choice. Draw a line from the midpoint of the hexagon to the midpoint of each side. These two radii are sixty degrees apart. The radius that has a slope nearest to the slope of the targeting line is the one that connects to the correct side. This algorithm will be implemented next month.

Functions `TgtShowPath()` and `GetVisibilty()` both traverse the targeting path, provided that the targeting line is not at a vertex angle. Why? Because a vertex angle implies that the targeting line passes straight between two adjacent hexagons. The targeting path then happens to be the same as the targeting line, so there is no point in drawing it. The visibility calculations, however, are more complicated. When a line of sight passes between two hexagons, then the total visibility is calculated as one-half the visibility of one hexagon plus one-half the visibility of the other. Support for this special case will be included in a future issue. In the meantime, `GetVisibilty()` returns negative one.

That wraps up this month. Thanks to all those who contributed both ideas and source code. With the next installment, you will be able to move a 'Mech and target from it. ♦

Peter Norberg, Consulting Consulting Services

Professional Solutions to Professional Problems

Peter Norberg, Consulting is a small software firm dedicated to providing its clients with superb service. We provide:

- * Conversion of DOS and "Windows" based packages to OS/2 2.0
- * Complete design and implementation of:
 - DLL Subsystems
 - Device Drivers
 - Complete Products
- * Debugging/rewrite of your code

PDNVDISK- The Two Gigabyte Ram Disk

Imagine the possibilities! The PDNVDISK software product allows you to make use of *all* of your memory for a RAM disk, even the memory which OS/2 is not willing to use for normal programs or swapping space. This allows you to use huge RAM drives for faster: voice mail, program development, read-only data bases, . . .

CONTACT: Peter Norberg, Consulting

(314) 647-8487

FAX: (314) 647-8528

OBJECT OBJECTIVE

SOURCES OF CLASS LIBRARIES

**DAVID
MOSKOWITZ**

There are three basic sources for class libraries. The first and most obvious is the language vendor. Second is the third-party vendor. The last source is the special library you create to meet your specific business needs.

Libraries from the language vendor are usually broad-based and potentially aimed more at a particular environment than solving business problems. Similarly, most of the third-party vendor libraries use the vendor libraries as a model and attempt to do a better job. While there is nothing wrong with this approach, it doesn't give sufficient help in solving real world business problems.

As a rule, the third party libraries offer more specialized classes than language vendors. However, vendor libraries, like their cousins from the language vendors, are usually language (and sometimes dialect) specific. If you're using Smalltalk, you buy Smalltalk classes. Further, you may have to buy class libraries for your brand of Smalltalk. Once you have the class library for language X, there is no easy way to either use or convert the library from one language (or dialect) to another.

To make matters worse, the current crop of class libraries have inconsistent quality. In addition, the style used to design and implement the class may also vary from one vendor to the next.

At some point, classes from different vendors should be able to work together; today, that isn't the case. Eventually, standards will replace the ad hoc approach that exists, now. In addition, the number of third party vendors will go up. Today, there are only a few, but the competition should increase rapidly.

In the early stages of OS/2 we talked about the potential for a market in dynamic link libraries (DLL's). That market never really materialized. However, today, there is a market for reusable classes. There are companies whose primary business is building and selling class libraries.

Rolling your own

Today, most companies that are doing object-oriented development start with libraries from either the language vendor or a third-party and add their own custom classes. Once we see more proven business classes available, this will change.

Regardless, what companies should do is use the general third-party libraries and begin to develop

specialized classes that are unique to the company's needs. These classes are not likely to be available in the market place, since these are the libraries that provide the competitive edge. These class libraries model the things that make your business special, that make it unique.

Building an object inventory

To get this competitive edge, the first step is to build an inventory of objects that model your business. Eventually, you want to move your software development effort out of program construction into object assembly.

To accomplish this goal, you will need to start by building models of the business process rather than trying to solve specific problems. Even when using an object-oriented approach, it is possible to fall into the classic software development life cycle: discover a problem, find a solution, implement the solution, and test and verify the results. That is the way we've done software development for nearly 50 years — even in the age of objects!

The problem with this approach to object-oriented development is that it potentially reduces the broad-based reusability of the resulting classes. Instead, we should begin to recognize that object-oriented development is more than just a new way to write software — it also represents a new way to think about the problem-solving process.

Instead of starting with a problem, start with the idea that the goal is to begin to build a model of some piece of the business. Design general-purpose high-level objects that provide the needed flexibility. Use overriding to provide specific operations at lower levels. Take advantage of the fact that object-oriented programming not only accommodates exceptions, it actually makes handling them much easier than other approaches.

Locating methods

We've talked about polymorphism and overloading, but we haven't really talked about how objects resolve a request. Recall that a message contains three parts: a receiver, a method name and whatever parameters may be required. When the object receives the message, it looks inside itself, first. If it finds the method, it executes it. If not, it looks up the hierarchy tree to its immediate superclass for the method (inheritance at work). Again, if the method is found, it is executed. This process con-

David Moskowitz is president of Productivity Solutions, 164 Avondale Road, Norristown PA 19403-2952. The consulting firm specializes in OS/2 and object-oriented development. He is the author of *Converting Applications to OS/2*, Brady Books, * 1989, and the forthcoming revision. David is a frequent speaker about OS/2 and object-oriented technology. He can be reached on CompuServe at 76701,100, on MCI Mail at 341-4084, and through the internet at 76701.100@compuserve.com

tinues up the tree until there is no place to go.

So, we want to define as many of the broad-based methods in the highest super classes that make sense. Remember, however, just because the method is defined at a high level does not mean it has to be fully implemented at that level. In fact, the high-level method definition will, in many cases be a routine to signal the caller that the receiver could not handle the request.

Keys to the kingdom

This is perhaps the hardest thing for new object-oriented programmers to grasp. What we want to be doing is to get out of the role of fire-fighting problem solving. Instead, use the six-step approach we suggested in a previous issue[1]: (a) pick a problem with a bit of urgency; (b) make the first attempt something that can be done in a reasonable amount of time; (c) pick something with a measurable impact; (d) pick a project that people can get passionate about; (e) make sure you won't need 14 levels of approval

just to get started; and (f) evaluate the results and learn from the experience.

The project should use a small team that starts by building a prototype. This prototype should be finished in a couple of weeks or, worst case, a couple of months. Once the prototype is done, the object is to refine the results until you have a deliverable.

The key difference between object-oriented software development and traditional software development, using an object-oriented approach, is that the prototype is not tossed. The prototype should be refined from the initial concept for a solution into a complete working model.

Once you get some practice with this approach, it will be faster than traditional software development. Further, this approach more easily accommodates the inevitable demand for changes that occur after the specifications are complete during the implementation phase.

No matter how good a job we do building specification, we normally have difficulty conceptualizing the end result until we be-

gin to see some progress — then the request for changes begins. Classic software development can be frustrating to both manager and developer alike. The manager wants changes and the developer has to find a way to kludge things to work the change in. The resulting effort is rarely as clean and bug-free as either party wants.

Change your approach

By using an object-oriented approach, a lot of this frustration can be avoided. Object-oriented development is a better way to approach the inescapable changes. Even so, we have seen very strong resistance to this approach.

This appears to be unstructured. The truth is, however, that the way software is supposed to be developed: specify, design, code, and test rarely happens in the real world. Very often companies will change the definition and requirements while the software is under development.

Rapid prototyping of projects versus long drawn-out formal developments is a more realistic approach to the software development problem.

Do not think we're suggesting that "hacking" is the way. Rather, this process is highly structured and disciplined; new solutions are constructed out of standard proven objects in your inventory. This same process can be used to create the stockpile of objects that will give your company a competitive edge, too.

The catch and the solution

This approach can lead to creeping "feature-itis". If the goal is a rapid prototype, it may never get done without a reality check. What is needed is a clear understanding of the basic requirements and a wish list. As you build the prototype, build the basics in, first. Then add things from the wish list until you either hit the end of a schedule or the end of phase dollars.

Once you get to this point, place a hard freeze on new features and functions and refine what has been completed until you have a working deliverable. Later, for the next version, you can add more things from your wish list. Besides, it is possible that things on your current wish list may become requirements for future versions.

References:

[1] Moskowitz, David, "Object Objective" in

Oberon Terminal Emulator/2

Version 1.21

Designed and Developed by / Terminal Emulator/2 is a Product of:
 Brady Flowers / Oberon Software, 518 Blue Earth Street
 Copyright 1992 by / Mankato, MN 56001. Call 1-507-388-7001
 Oberon Software / 24 Hour a day BBS line at 1-507-388-1154

✓ XModem, XModem-1K, YModem, YModem-G, ZModem, CISM B-Plus
 ✓ ANSI Terminal, VT100, IBM3101, Standard TTY
 ✓✓ Low \$65.00 Price ✓✓ Free technical support!

Oberon Software
 518 Blue Earth St.
 Mankato, MN 56001

For information call:
 1-507-388-7001
 Credit card orders call PSL at:
 1-800-242-4775

IN THE TRENCHES

I prepared carefully for the Windows and OS/2 conference in Boston. I checked my weapons. My Uzi was fully loaded, and I was looking forward to confronting Microsoft at their challenge booth. Unfortunately, they didn't show up. I wonder if they knew I was waiting for them, or did they catch so much flak about their challenge booth that they wanted to use Win NT vs OS/2 instead? Of course, the official line is "this conflicts with our national sales meeting". Note, this is the same excuse they used last year. Of course, the reality is probably that NT is nowhere near ready for prime time, and they didn't want to be embarrassed. Maybe somebody at MS can explain this to me.

Still, the no-show by Microsoft didn't dampen my enthusiasm. However, I did manage to dampen the enthusiasm of one unfortunate IBM'er. It seems that when I flushed my toilet, the damned thing didn't shut off and flooded the clothes closet of the occupant of the hotel room below mine (Sorry, Terry!). Still, he took it all in stride, especially since the Marriott picked up the bill for dry cleaning.

Lots of enthusiasm was to be found at the OS/2 Emporium, where one IBM'er spotted a software developer wearing a T-shirt he got at the NT developers conference in July. After selling him on various IBM goodies (coffee mug, gym bag, etc.) he made one last offer: "Give me the NT shirt and I'll give you an OS/2 shirt." He took the offer. The "Weasel Pelt" (as it was called) was then displayed to any and all comers.

Meanwhile, on the software front, lots of interesting things were happening. Stac Electronics was showing Stacker for OS/2. The product is slated to start beta testing in September, with an expected release before the end of the year. Initially, only FAT partitions will be supported, as there is a performance penalty associated with using HPFS partitions. Hopefully, Stac will fix this soon.

Hilgraeve's new GUI version of their HyperAccess/5 communications software is scheduled to beta test, with a tentative release date of November (sounds like another one for Fall COMDEX).

Quark Research was showing their PDQ relational database, which is capable of sorting some 1-4 million records per minute! Pricing ranges from \$595 and up (depending on the number of workstations on a LAN).

I saw a brief demo of a really neat software development tool called Parts (actually, the full title is "Parts Workbench for OS/2"). Parts is an object-oriented program that lets you link various dials, gauges, levers, buttons and so on to each other, creating a sophisticated control panel. Unfortunately, it was left running on a machine by itself, so I never even got to ask questions about the author. But I did get an opportu-



nity to receive a review copy and my review appears earlier on in the magazine.

One Up Corporation had a fascinating screen-saver program (I think the name was squeegee). Basically, it would pour water on the screen, which would run off like droplets off of a window. Then an automated squeegee would come along and clean off the mess. I want one!

There was some good news in the hardware department. Both MediaVision (makers of the PAS-16 card) and Ad Lib (Makers of the SBPro) seem to be engaged in a war to see who gets the lion's share of the OS/2& MCA markets. Ad Lib is shipping an

MCA version of their board, and MediaVision is to follow suite. Both companies are racing to get out drivers for their sound cards (sound for which you owners of MMPM/2 won't have much longer to wait).

Best quote of the show goes to Paul Pignatelli of the Corner Store, for his multimedia price list, labelled "Appetizers - Hardware" done up like a menu.

Speaking of hardware, IBM was showing its ThinkPad computer system, running pen extensions to OS/2. Lois Dimpfel did the honors. Both sound and images were able to be sent on a token ring network to another ThinkPad (a good demo of some promising technology). The technology would really be neat if they could actually figure out how to get some decent hard disk storage into one of these things without having to run around with a 50' cord connected to a hard drive.

IBM's Lee Reising gave an overview of how DOS, OS/2, and AIX have overlapping areas. And yes, he briefly talked about Portable OS/2 (IBM's version of what NT should be). It's still tough to figure out if Taligent is really going to be a separate OS, or the technology merely used to enhance existing OS/2 and System 7 software.

IBM was also busy giving out awards. The millionth copy of OS/2 was given out to Caterpillar Corporation. In honor of this, IBM had a huge set of dominoes laid out in a line. Once started, it took 3 or 4 minutes for the dominoes all to fall. Lots of media people were present, including TV crews.

IBM also unveiled its trophies to be awarded to software developers for writing OS/2 applications. Corel Draw, PM Draw, PFS:Works and DCF/2 were given awards. Also a giant trophy listing some 200 names was unveiled, to be reserved for the first 200 32 bit apps. This huge (4' tall) trophy will be on display at future computer shows.

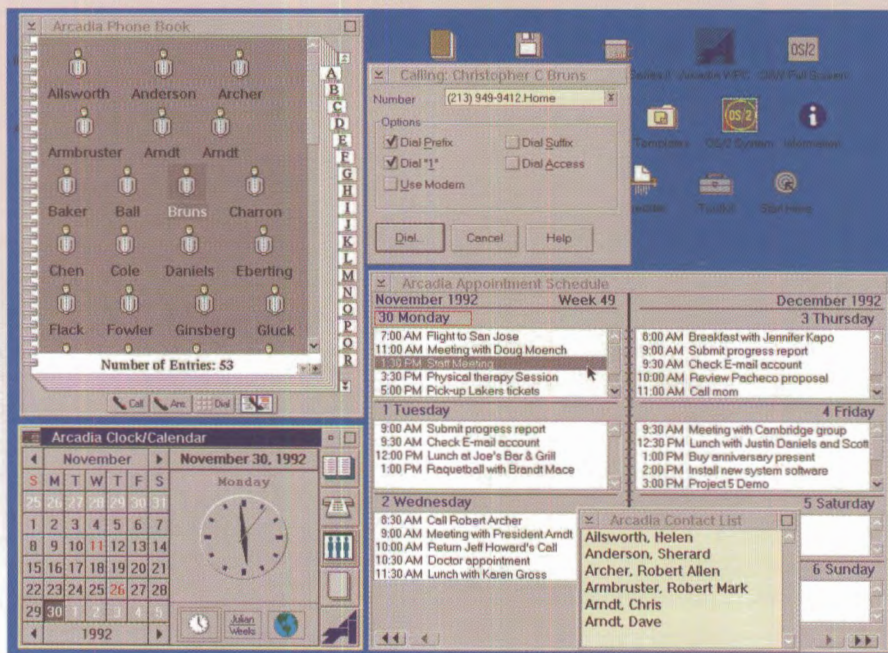
All in all, it was a good show. I can hardly wait for Fall COMDEX.

ERIC PINNELL

Arcadia Workplace Companion™

An **OS/2** Personal Information Manager

NEW



OS/2

The Arcadia Workplace Companion (WPC) is a set of productivity tools designed specifically for the OS/2 Workplace Shell environment from the ground up! It adheres to Common User Access (CUA) '91 guidelines and completely integrates its various components in an object-oriented fashion. The WPC exploits all of the powerful new controls found within OS/2 2.0 including notebooks, containers, sliders, value sets, and spin buttons.

Everything necessary to schedule the day's events, keep track of appointments, and manage business contacts is represented graphically. The Workplace Companion contains a clock/calendar module, an appointment book, a telephone/address book (with phone log), a notepad, a to-do list, and other productivity features.

Finally, an
Intuitive personal
Information
management tool
for
OS/2 2.0!

Clock/Calendar. This module contains a month calendar, a full-year calendar, Julian date, analog/digital clock, international times, holidays, and more! It allows you to customize its many features, such as setting your own alarms and adding new cities to the international cities list.

Appointment Book. The Appointment Book is an excellent way to schedule your day's events. Appointments may be added by clicking on the time you wish to schedule and typing a description.

Telephone/Address Book. This module is a convenient way to store the name, numbers, and addresses of all your important contacts. The built-in Phone Log and Follow-up feature make tracking phone calls easy.



**Arcadia
Technologies
Inc.**

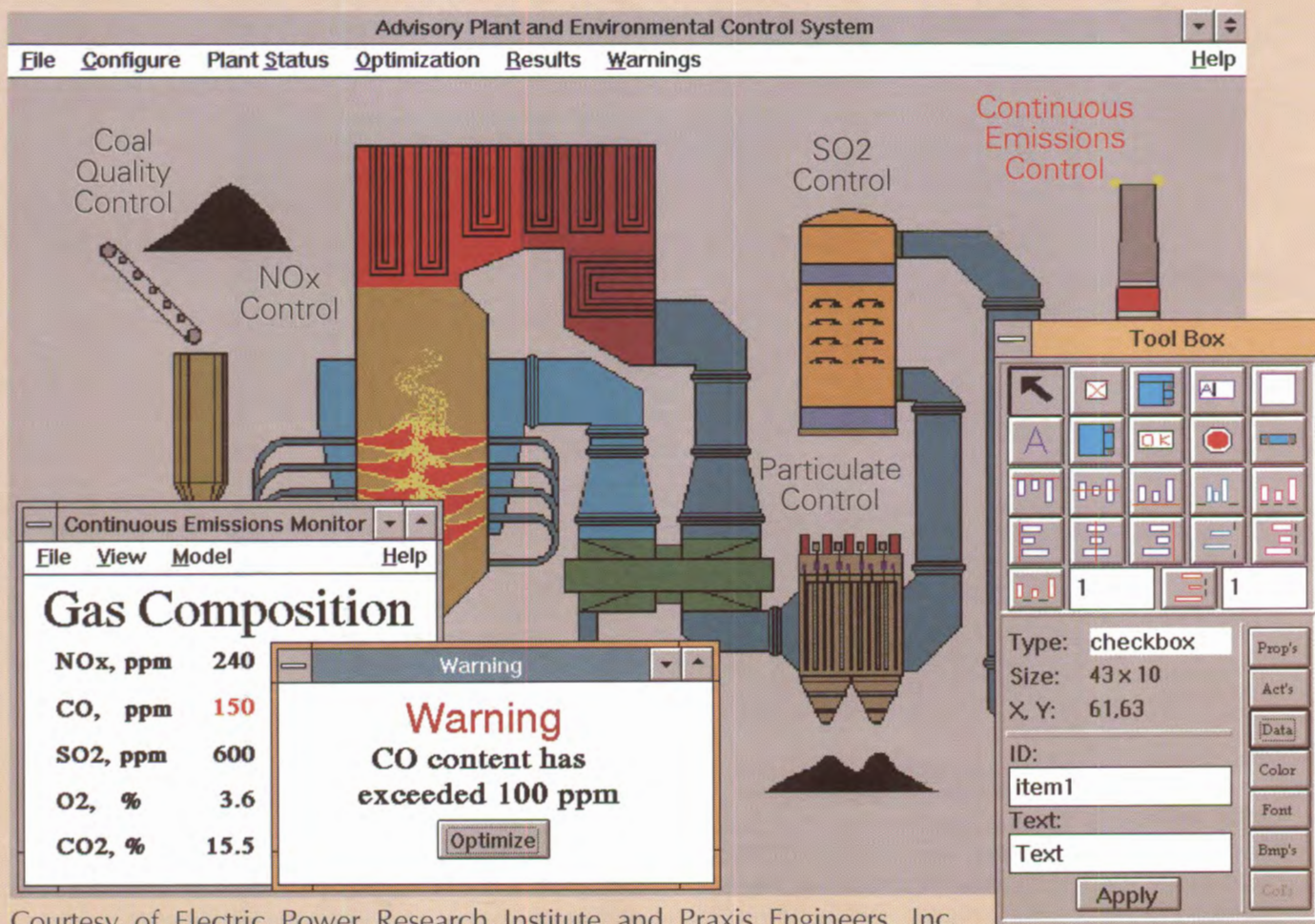
OS/2, CUA, IBM are registered trademarks of International Business Machines Corporation. Arcadia Workplace Companion is a trademark of Arcadia Technologies, Incorporated.

735 West Duarte Road, Suite 207
Arcadia, CA 91007

Tel: (818) 446-6945
Fax: (818) 447-4212

Special
Introductory Offer
\$79.95

GUI Tool for Professional Developers



Courtesy of Electric Power Research Institute and Praxis Engineers, Inc.

So you'd like to develop serious applications, but you haven't found your ideal GUI tool for OS/2?

GUILD is for YOU!

- **Productive.** Point & click screens, actions and data.
- **Flexible.** Open architecture for custom C/C++ coding.
- **Portable.** Port across Windows, OS/2, Motif and Mac.

(415) 593-3200
Phone

(415) 595-8158
FAX

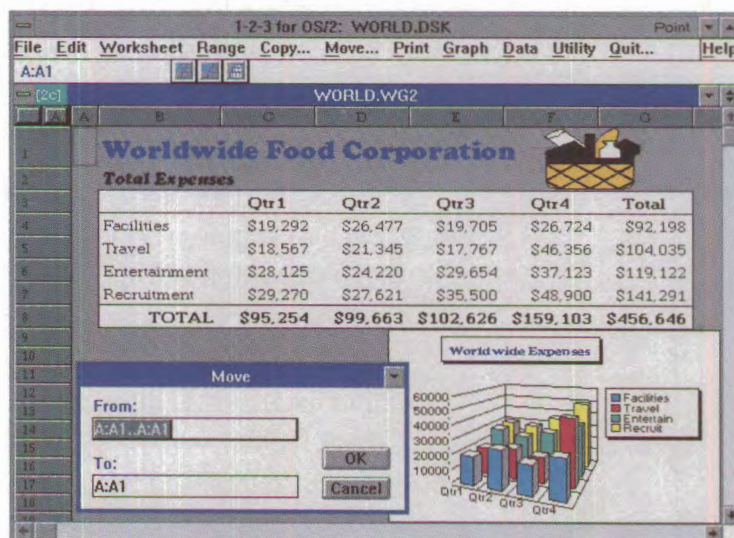


1301 Shoreway Road, Belmont, Calif. 94002

Trademarks and registered trademarks belong to their respective holders.

Getting the most out of OS/2 is as easy as 1-2-3.

Nothing exploits the full graphical environment of OS/2® like Lotus® 1-2-3® for OS/2. It's the only spreadsheet that takes full advantage of the speed, memory and multi-tasking capabilities of OS/2, while offering all the power and functionality you've come to expect from 1-2-3. This includes Solver and Backsolver, true 3D worksheets, file-linking and access to external databases. As well as such industry-leading features as WYSIWYG (What-You-



Thanks to its multi-windowing capabilities, 1-2-3 for OS/2 lets you look at your graphs and spreadsheet simultaneously. Whether it's working with a 3D file or bringing in information from external databases, the full power of 1-2-3 for OS/2 is always within reach.



See-Is-What-You-Get) and Collections, which allows you to work with multiple cells and ranges simultaneously.

1-2-3 for OS/2 is also the only spread-

sheet that's fully compatible with IBM® OS/2 version 2.0. And of course, it's compatible with all previous versions of 1-2-3, across all platforms.

For a free demo disk, including sample spreadsheets to show you just how powerful 1-2-3 for OS/2 is, call **1-800-TRADEUP, Ext. 6565.**

1-2-3 for OS/2

© 1992 Lotus Development Corporation. All rights reserved. Lotus and 1-2-3 are registered trademarks of Lotus Development Corporation. IBM and OS/2 are registered trademarks of International Business Machines Corporation.

Let Gpf write the GUI you design



SYSTEMS, INC.

Using the powerful point and click visual programming environment of Gpf*, you can prototype, test and generate a complete OS/2 PM GUI in a few hours or days rather than the weeks or months required to hand code the same design. Even a relatively simple GUI can require writing thousands of lines of code, but with Gpf you simply draw your user interface on the screen. The integrated dialogue editor of Gpf permits actions and context sensitive help to be linked to controls as you create them. Gpf then generates error free ANSI C complete with embedded SQL statements.

Gpf is optimized to take full advantage of OS/2 PM, the most powerful and robust GUI system available. Since Gpf code directly accesses the PM API, there is no run time module to distribute with your application and no added overhead or royalties.

Gpf keeps the entire design definition in one file. This means single point maintenance for easy update and archiving. From this file Gpf generates the C source file as well as .H, .RC, .IPF, .DEF, .IDS, .MAK, etc..

Gpf supports:

- Simple and direct linkage of the interface to program logic, built in or user defined functions.
- Direct association of help screens with controls, and complete integration into the PM Help Presentation Facility.
- Flexible use of Presentation objects (fonts, colors, etc.) with controls and windows (client area and frame).
- Simple inclusion of bitmaps for use on About screens, user-defined buttons, and menu or pulldown entries.
- Automatic embedded SQL statements to read OS/2 DataBase Manager tables, directly into combo or list boxes.
- Multi-thread programming.
- Multiple source file generation.
- Automatic creation of controls that scale with window size.
- Inclusion of user defined controls

Try us out.

Order Gpf today for just \$995.⁰⁰

Call **Gpf Systems Inc.** at:

(203) 873-3300 or (800) 831-0017 - fax (203) 873-3302.

Free demo software available

P.O. Box 414, 30 Falls Rd., Moodus, CT 06469

* GUI Programming Facility